الفصل الأول

كتابة أول برنامج لك في فيجول بيسك

سنكتب في هذا الفصل، أول برنامج لك في فيجول بيسك. تمر مرحلة كتابة البرنامج في فيجول بيسك بخطوتين:

- خطوة التصميم المرئي للبرنامج.

سندعوها عبر الكتاب بالتمثيل المرئي Visual Implementation للنموذج.

- خطوة كتابة نص البرنامج.

يُصممِ المستخدم خلال الخطوة الأولى، البرنامج باستخدام الأدوات التي تأتي مع برمجية فيجول بيسك. تمكّنك هذه الأدوات من تصميم البرنامج باستخدام الفأرة ولوحة المفاتيح.

لا يلزمنا خلال مرحلة البرمجة المرئية (أي التصميم المرئي) كتابة أي نص برمجي!، وكل ما عليك معرفته هو كيف تشغّل وتستخدم الأدوات البرمجية التي تأتي مع فيجول بيسك. وسوف تلاحظ أن عملية التصميم المرئي فيها قدر كبير من المتعة وتعتمد بكثرة على النقر بالفأرة. يركز هذا الفصل على معرفة كيفية استخدام أدوات التصميم المرئي في فيجول بيسك.

أما في خطوة كتابة نص البرنامج، فيستخدم محرر نصوص لكتابة البرنامج. وتتألف البرامج من عبارات مكتوبة بلغة البرمجة فيجول بيسك. تتشابه عملية كتابة نصوص البرامج في فيجول بيسك مع كتابة البرامج في اللغات الأخرى. إلا أن كتابة البرامج في فيجول بيسك أسهل بكثير من كتابتها باللغات الأخرى.

حول هذا الفصل

ستلاحظ إذا استعرضت باقي فصول الكتاب، أن هذا الفصل ليس نموذجياً، فهو يركز على ناحية البرمجة المرئية Visual Programming للغة فيجول بيسك، ولهذا فهو يدقق على كيفية استخدام الأدوات البرمجية للغة (عناصر التحكم). بينما تتولى فصول الكتاب المتبقية تعليمك، كيفية كتابة نص البرامج في فيجول بيسك.

إنشاء دليل حفظ الملفات

قبل البدء بعملية كتابة أول برنامج لك في لغة فيجول بيسك، سننشئ دليلاً يحوي على ملفات العمل المنجز. وسنفترض عبر هذا الفصل، أن الدليل موجود لديك على القرص الصلب، وسنطلب منك فقط حفظ الملفات فيه، لهذا أنشئ الدليل التالي C:\VB5Prg\Ch01.

برنامج الترحيب

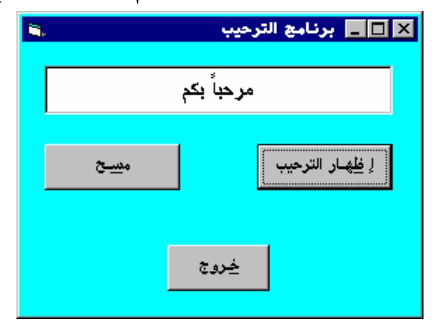
سنكتب برنامجاً يدعى برنامج الترحيب. وقبل كتابة برنامج الترحيب بنفسك دعنا ندرس أولاً مواصفاته، وبهذه الطريقة سنتمكن من فهم ما يفترض أن ينجزه هذا البرنامج بشكل أفضل.

يظهر الإطار المبين في الشكل 1-1، عند بدء تشغيل برنامج الترحيب وكما تلاحظ يحوي إطار البرنامج على ثلاثة أزرار أوامر (الزر **إظهار الترحيب** والزر **مسح** والزر **خروج**) ومربع نص فارغ.



الشكل 1-1
برنامج الترحيب.

تظهر الرسالة **مرحباً بكم** ضمن مربع النص، عند النقر على الزر **إظهار الترحيب**.



الشكل 1-2
إظهار مرحباً بكم
في مربع النص.

يمحو برنامج الترحيب محتوى مربع النص، عند النقر بالفأرة على الزر **مسح**.

ينتهي تنفيذ برنامج الترحيب عند النقر على الزر **خروج**.

إنشاء مشروع جديد

الآن وقد علمت ما ينجزه برنامج الترحيب، نستطيع الشروع بكتابته.

ملاحظة
برنامج الترحيب عبارة عن برنامج بالغ البساطة، لكن مع ذلك يتوجب عليك كتابته بنفسك، لأنه يمثل برنامجاً نموذجياً في فيجول بيسك. وفي الواقع، حال تعلمك كيفية كتابة برنامج الترحيب بنفسك، تستطيع فهم ما هي لغة فيجول بيسك! طبعاً هنالك كمّ هائل من المعلومات الأخرى في فيجول بيسك لا بد لك من معرفتها، إلا أن كتابة هذا البرنامج بنفسك يعني أنك تعرفت على أساسيات فيجول بيسك.

أُولى الأوليات التي يجب عليك إنجازها، هي إنشاء مشروع جديد New Project لبرنامج الترحيب باتباع الخطوات التالية:

- شغّل فيجول بيسك. إذا رأيت مربع الحوار المبين في الشكل 1-3، فأغلق هذا المربع بنقر الزر **إلغاء الأمر**.
- اختر البند **New Project** من القائمة **File** لفيجول بيسك.

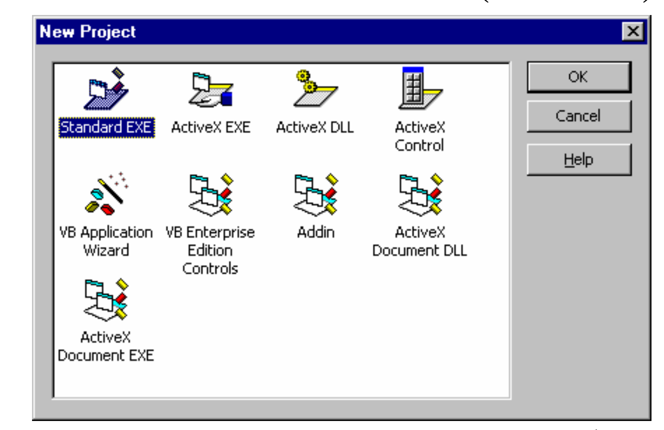
يستجيب فيجول بيسك بإظهار مربع الحوار New Project المبين في الشكل 1-4.



الشكل 1-3

مربع الحوار New Project الذي قد يظهر عند تشغيل فيجول بيسك.

كما يبدو من الشكل 1-4. يمكّنك مربع الحوار New Project من إنشاء أنواع مختلفة من المشاريع (Project). لكننا في الوقت الراهن، نرغب بإنشاء تطبيق تنفيذي قياسي (Standard EXE).



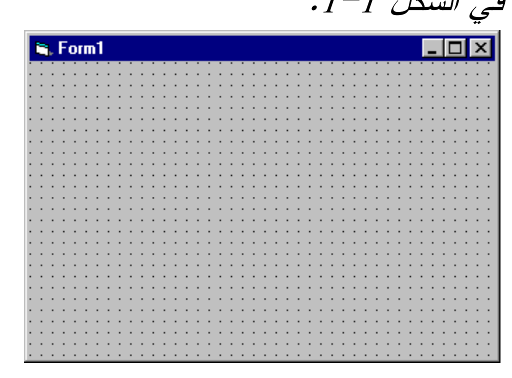
الشكل 1-4

مربع الحوار New Project الذي يُظهره فيجول بيسك بعد اختيار **New Project** من القائمة **File**.

لهذا أَخبر فيجول بيسك بذلك باتباع الخطوة التالية:

- انقر الرمز Standard EXE الذي يظهر ضمن مربع الحوار New Project ثم انقر الزر **OK** لمربع الحوار New Project.

يظهر عند ذلك إطار خال يدعى النموذج Form1 (انظر الشكل 1-5). سنستخدم أدوات لغة فيجول بيسك المتنوعة حتى يصبح النموذج الخالي، كالمبين في الشكل 1-1.



الشكل 1-5

النموذج Form1 الفارغ.

حفظ المشروع الجديد

رغم أننا لم نجر أي تعديلات بعد على النموذج الفارغ، لكن يتوجب عليك حفظ المشروع في هذه المرحلة المبكرة من التصميم. يؤدي حفظ المشروع إلى تخزين ملفين:

- ■ملف المشروع Project File، ويمتلك الامتداد vbp. يحتوي هذا الملف على المعلومات التي يستخدمها فيجول بيسك لبناء المشروع.
- ■ملف النموذج Form File، ويمتلك الامتداد frm. ويحتوي على معلومات تتعلق بالنموذج نفسه.

استخدم الآن الخطوات التالية لحفظ الملفين: Hello.vbp (ملف المشروع)، و Hello.frm (ملف النموذج).

□تحقق من تمركز الإضاءة عند عنوان الإطار Form1، ثم اختر **Save Form1 As** من القائمة **File** لفيجول بيسك. (يعني تمركز الإضاءة عند نموذج ما، أنه تم اختياره).

يستجيب فيجول بيسك بإظهار مربع الحوار Save File As.

□استخدم مربع الحوار Save File As لاختيار الدليل C:\VB5Prg\Ch01 من أجل حفظ الملف فيه. بدّل الاسم الافتراضي للنموذج من Form1.frm إلى Hello.frm (انظر الشكل 1–6).

□انقر الزر **حفظ** في مربع الحوار Save File As.

يستجيب فيجول بيسك بحفظ النموذج باسم Hello.frm في الدليل C:\VB5Prg\Ch01.

ملاحظة

لا تستخدم الاسم الافتراضي الذي يقدمه فيجول بيسك عند حفظ نموذج ما. بل احفظ النموذج باسم مناسب للتطبيق الذي تصممه. لاحظ مثلاً، أننا استخدمنا الاسم Hello.frm كاسم لنموذج برنامج الترحيب.




الشكل 1-6
حفظ النموذج
بالاسم Hello.frm.

والآن، احفظ ملف المشروع:

- اختر **Save Project As** من القائمة **File** لفيجول بيسك.

يستجيب فيجول بيسك بإظهار مربع الحوار Save Project As.

- الاسم الافتراضي الذي يقدمه فيجول بيسك للمشروع هو Project1.vbp.

لكن لا بد من تغيير الاسم الافتراضي إلى اسم يتناسب مع التطبيق الذي تعمل على تطويره.

- استخدم مربع الحوار Save Project As لحفظ المشروع بالاسم Hello.vbp في الدليل C:\VB5Prg\Ch01.

ملاحظة

لا تستخدم الاسم الافتراضي الذي يعطيه فيجول بيسك للمشروع عند حفظ ملف المشروع. ولكن بدلاً من ذلك أطلق على المشروع اسماً يتناسب مع التطبيق الذي تصممه. لاحظ مثلاً أننا أطلقنا على مشروع برنامج الترحيب تسمية Hello.vbp. نكون حتى هذه اللحظة قد أنهينا حفظ الملفين Hello.vbp (ملف المشروع) وHello.frm (ملف النموذج).

فحص إطار المشروع Project Window

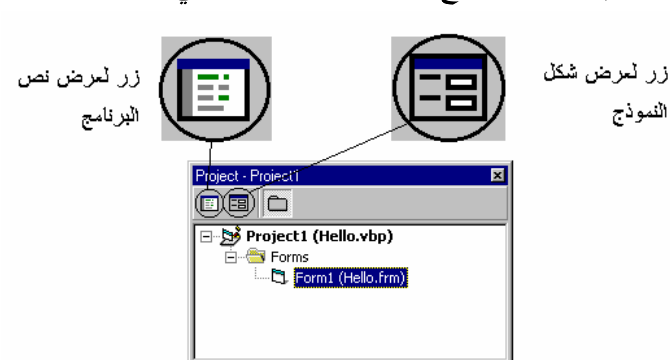
حتى هذه النقطة، يدعى المشروع بالاسم Hello.vbp ويتألف من ملف نموذج واحد هو الملف Hello.frm. سنمر عبر الفصول القادمة على مشاريع تحوي أكثر من ملف نموذج.

يعتبر إطار المشروع Project Window من الأدوات الهامة التي يقدمها فيجول بيسك، ويمكّنك من رؤية الملفات المختلفة الموجودة في المشروع. (ستشعر بقيمة هذه الميزة بشكل أفضل كلما زاد تعقيد المشروع).

اتبع الخطوات التالية لفهم محتويات إطار المشروع Project Explorer:

- اختر **Project Explorer** من القائمة **View** لفيجول بيسك.

يظهر إطار المشروع حسب ما هو مبين في الشكل 1-7.



الشكل 1-7
إطار نافذة المشروع
.Project Window

ملف المشروع، حسب ما يظهر من إطار المشروع هو Hello.vbp. وهو يحتوي على نموذج وحيد باسم Hello.frm. يعتبر المشروع الحالي بسيط جداً، بحيث لا تغدو أهمية إطار المشروع واضحة للعيان. لكن مع تزايد تعقيد المشاريع التي ستكتبها مستقبلاً، سوف تلاحظ مدى أهمية هذا الإطار.

تغيير الخاصية Caption للنموذج (تغيير عنوان النموذج)

اتفقنا أن النموذج الفارغ الذي أنشأه فيجول بيسك يحمل العنوان Form1 (انظر الشكل 1-5). هذا العنوان يمثل العنوان الافتراضي الذي يعطيه فيجول بيسك للنموذج الفارغ عند إنشائه. تستطيع ترك هذا العنوان على حاله، ولكننا نفضل أن نطلق عنواناً على

النموذج يكون مناسباً لموضوعه. فمثلاً حسب الشكل 1-1، يحمل النموذج عنوان برنامج الترحيب، وفي هذا دلالة على أن البرنامج هو برنامج ترحيب.

ملاحظة

النموذج هو إطار (Window) بنفس الوقت. فمثلاً، النموذج Form1 المبين في الشكل 1-5 يدعى نموذج ويدعى إطار (Window) بذات الوقت. يطلق مصطلح النموذج، على النافذة التي أنشأتها في مرحلة التصميم، ويستخدم مصطلح الإطار (أو نافذة)، عند تنفيذ البرنامج. بكلام آخر، عند ذكر مصطلح النموذج، يكون المقصود مرحلة التصميم، وعند ذكر مصطلح الإطار أو النافذة، يكون المقصود مرحلة التنفيذ.

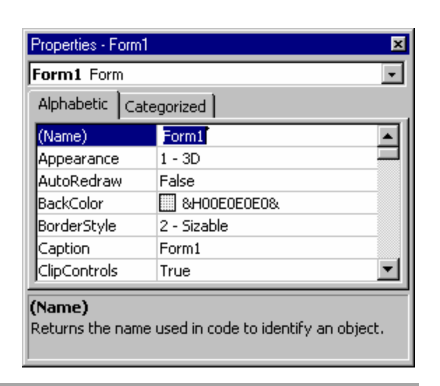
نبين لك الآن طريقة تغيير عنوان النموذج الفارغ لبرنامج الترحيب:

- تحقق من اختيار النموذج الفارغ. تستطيع التأكد بسهولة من اختيار نموذج، بتفحص شريط عنوانه. فإذا كان شريط العنوان (Caption) مضاءً، فهذا يعني أن النموذج تم اختياره. أما إذا لم يكن قد تم اختيار النموذج، فيكفي النقر النموذج في أي مكان على سطحه لاختياره.

أو تستطيع بدلاً من ذلك اللجوء إلى إطار المشروع Project Window (اختر **Project Explorer** من القائمة **View** لفيجول بيسك)، ثم اختيار العنصر Hello.frm بنقر الفأرة عليه، ثم نقر الرمز View Object الذي يظهر أعلى إطار المشروع. (انظر الشكل 1-7. يظهر الرمز View Object أعلى نافذة المشروع وثاني رمز من جهة اليمين).

بعد اختيار النموذج، اختر إطار الخصائص **Properties Window** من القائمة **View** لفيجول بيسك.

يستجيب فيجول بيسك بإظهار إطار الخصائص (Properties Window)، الشكل 1-8.



الشكل 1-8

إطار الخصائص

Properties Window.

ملاحظة

يمكنك في فيجول بيسك نقل شتى الإطارات إلى أي موقع ضمن سطح مكتب فيجول بيسك وذلك بسحب عناوين هذه الإطارات بواسطة الفأرة.

قد يتغير شكل الإطارات قليلاً تبعاً للموقع الذي تأخذه على سطح المكتب. فمثلاً قد يتغير شكل إطار الخصائص Properties Window إلى حد ما، عما هو مبين في الشكل 1-8، بحسب الموقع الذي يحتله على سطح المكتب، لكن مهما كان موقعه، تستطيع التعرف عليه، بسبب احتوائه على كلمة الخصائص Properties في عنوانه.

- انقر على الخلية التي تظهر يمين اسم الخاصية Caption (خلية العنوان) في إطار الخصائص (Properties Window).

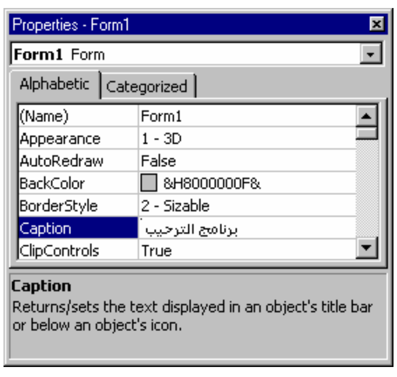
الآن ستجد، أن الخلية الواقعة يمين الخلية Caption تملك النص Form1.

- استخدم المفتاح Delete ومفاتيح الأسهم على لوحة المفاتيح، لحذف النص Form1، واستبدله بكتابة النص **برنامج الترحيب**.

يفترض أن يبدو إطار الخصائص كما في الشكل 1-9.

تهانينا! لقد أنهيت للتو عملية تبديل الخاصية Caption (العنوان) للنموذج.

ألق نظرة على النموذج الفارغ (انظر الشكل 1-10). يحمل هذا النموذج الآن العنوان **برنامج الترحيب**.



الشكل 1-9
تبديل الخاصية Caption للنموذج.



الشكل 1-10
النموذج frmHello وهو يحمل الآن عنواناً جديداً.

ما هي الخاصية ؟!

الخاصية Caption ما هي إلا إحدى خصائص النموذج، فكما تشاهد من إطار الخصائص Properties Window، فإن النموذج يمتلك الكثير من الخصائص الأخرى. ولفهم معنى الخاصية لا بد لك من فهم كيفية تعامل فيجول بيسك مع الكائنات Objects مثل النماذج Forms وأزرار الأوامر Command Buttons وأشرطة التمرير Scroll bars ومربعات الاختيار Check Boxes . . . الخ.

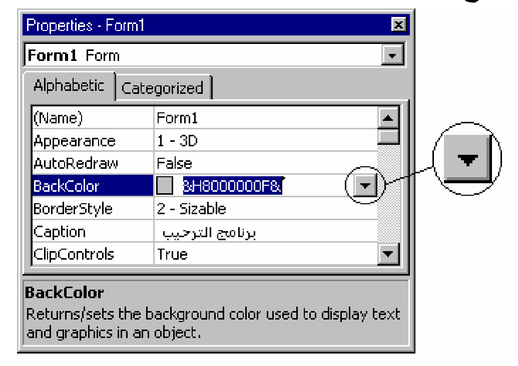
تُعرّف خصائص الكائن (Object) كيف يبدو هذا الكائن وكيف يتصرف. فمثلاً النموذج عبارة عن كائن. تحدد الخاصية Caption للنموذج، النص الذي يظهر في شريط عنوانه.

خذ مثلاً الخاصية BackColor للنموذج، تحدد هذه الخاصية لون خلفية النموذج. اتبع الخطوات التالية لتغيير الخاصية BackColor للنموذج:

- تحقق من اختيار النموذج. (انقر أي مكان من النموذج لاختياره).
- اختر **Properties Window** من القائمة **View** لإظهار إطار الخصائص.

□انقر على الخلية الواقعة يمين الخلية BackColor في إطار الخصائص.

يضع فيجول بيسك عند نقر هذه الخلية، رمز سهم نازل فيها، (انظر الشكل 1-11).

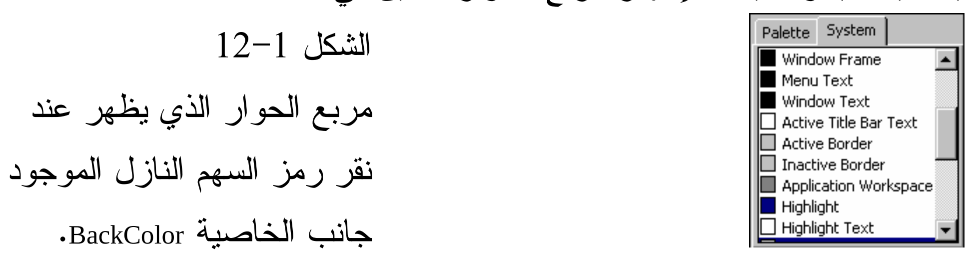


الشكل 1-11

الخاصية BackColor.

□انقر رمز السهم النازل الذي يظهر في الخلية.

يستجيب فيجول بيسك بإظهار مربع الحوار المبين في الشكل 1-12.



الشكل 1-12

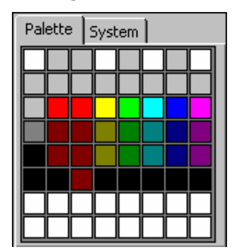
مربع الحوار الذي يظهر عند نقر رمز السهم النازل الموجود جانب الخاصية BackColor.

لاحظ أن مربع الحوار المبين في الشكل 1-12 يمتلك صفحتين:

الصفحة Palette، والصفحة System. الصفحة التي تظهر وفق الشكل 1-12 هي الصفحة System.

□انقر على الصفحة Palette في مربع الحوار المبين في الشكل 1-12.

يستجيب فيجول بيسك بإظهار صفحة الألوان Palette (انظر الشكل 1-13).



الشكل 1-13

الصفحة Palette

صفحة الألوان.

□ اختر اللون الذي تحبذه بنقره في الصفحة Palette. لنفترض الآن أن اللون الذي اخترته هو اللون الذي يظهر في المربع الواقع عند العمود الثالث والسطر الثالث من

جهة الأعلى. (أو اختر أي لون تفضل).

ملاحظة

تفحص الخلية التي تقع يمين اسم الخاصية في نافذة الخصائص والتي ترغب بتعيينها. فإذا كانت تلك الخلية تحمل رمز سهم نازل داخلها، أو زر يحوي ثلاث نقاط متجاورة. انقر على السهم أو الزر، فيظهر إطار آخر أو لائحة، تمكنك من اختيار قيمة ما بواسطة الفأرة للخاصية المحددة.

جرب عدة ألوان بتكرار العملية، إلى أن تشعر بالرضا عن اللون الذي تختاره.

تبديل الخاصية Name للنموذج (اسم النموذج البرمجي)

يجب أن يمتلك كل كائن (Object) في فيجول بيسك اسماً، يتحدد ذلك الاسم بواسطة الخاصية Name لذلك الكائن.

فمثلاً، عندما أنشأنا النموذج الجديد لبرنامج الترحيب، أطلق فيجول بيسك من تلقاء نفسه الاسم Form1 على النموذج (أي أسند الاسم Form1 إلى الخاصية Name لنموذج برنامج الترحيب).

ملاحظة

لا تخلط بين الخاصية Caption، والخاصية Name للنموذج. تستخدم الخاصية Caption لإظهار عنوان ما، في شريط عنوان النموذج. أما الخاصية Name فتستخدم لإسناد اسم برمجي خاص بالنموذج نفسه. مع أن فيجول بيسك يفترض القيمة Form1 لكلتا الخاصيتين عند إنشاء نموذج جديد فارغ.

غيّر الآن الخاصية Name للنموذج:

- انقر على النموذج في أي مكان منه لاختياره.
- اختر **Properties Window** من القائمة **View**.

يستجيب فيجول بيسك بإظهار إطار خصائص النموذج Form1.

يملك إطار الخصائص Properties صفحتين هما: الصفحة Alphabetic والصفحة Categorized (انظر الشكل 1−8). عند اختيار الصفحة Alphabetic، تُرتب الخصائص أبجدياً (باستثناء أهم خاصية وهي Name التي تظهر أولاً). بينما تظهر الخصائص مصنفة حسب مواضيعها، عند اختيار الصفحة Categorized.

□اختر الصفحة Alphabetic لإطار الخصائص.

تظهر الخاصية Name في مقدمة لائحة الخصائص.

□انقر الخلية التي تظهر يمين الخاصية Name، يمكّنك فيجول بيسك من تعديل الخاصية Name.

□استبدل الاسم الافتراضي Form1 بالاسم frmHello.

غيّرنا في الخطوة السابقة الخاصية Name للنموذج من Form1 إلى frmHello. تشير الأحرف الثلاث الأولى من قيمة الخاصية Name للكائنات Objects إلى نوع الكائن.

وهكذا فالأحرف الثلاث الأولى من الخاصية Name لنموذج ما، هي frm، كما في مثالنا الحالي frmHello.

ملاحظة
غيّر الأسماء الافتراضية للكائنات بحيث تعكس أسماؤها ووظائفها في البرنامج. فمثلاً frmHello عبارة عن اسم النموذج الذي يستخدم من قبل برنامج الترحيب. يؤدي بدء اسم النموذج بالأحرف الثلاث frm إلى تسهيل فهم البرنامج وتصحيحه. ألق نظرة على الاسم frmHello، بما أنه يبدأ بالأحرف frm. تستطيع بسهولة (أنت أو من يقرأ الاسم) أن تعلم مباشرة أن نوع الكائن frmHello عبارة عن نموذج. لا يعتبر هذا الأمر من متطلبات البرمجة في لغة فيجول بيسك، لكنه كما قلنا يسهل قراءة وفهم البرنامج.

تبديل الخاصية RightToLeft للنموذج (تعريب النموذج)

يمكّنك فيجول بيسك، من إنشاء برامج عربية المظهر والمضمون، بشرط تطوير برنامجك في نظام تشغيل يدعم اللغة العربية، مثل Windows95 العربي أو WinNT الداعم للغة العربية.

وما عليك سوى تغيير قيمة الخاصية RightToLeft من القيمة False إلى القيمة True، لأي كائن من كائنات فيجول بيسك، حتى يظهر الكائن بشكل مقبول للمستخدم العربي.

ملاحظة
للحصول على معلومات كاملة، عن موضوع إنشاء التطبيقات العربية في فيجول بيسك، اقرأ الفصل الثاني والعشرين (إنشاء تطبيقات عربية السمة مع فيجول بيسك 5). وكل ما يهمنا معرفته الآن، أن الخاصية RightToLeft = True تعني تعريب الكائن.

غيّر الآن الخاصية RightToLeft للنموذج:

- انقر على النموذج في أي مكان منه لاختياره.
- اختر **Properties Window** من القائمة **View**.

يستجيب فيجول بيسك بإظهار إطار خصائص النموذج Form1.

- انقر خلية الخاصية RightToLeft نقراً مزدوجاً، لتغيير قيمتها من القيمة False إلى القيمة True.

يستجيب فيجول بيسك، بتعريب النموذج بأن يظهر عنوانه على يمين المستخدم، بدلاً من يساره.

أصبح النموذج الآن، نموذجاً عربياً. وعند إضافة أي عنصر تحكم (أداة من أدوات فيجول بيسك)، سيقوم فيجول بيسك بإسناد القيمة True للخاصية RightToLeft للعنصر الجديد آلياً.

حفظ العمل المنجز

لم ننته بعد من النموذج (تذكر أن النموذج سيبدو عند انتهائه كما في الشكل 1–1)، لكن رغم ذلك يفضل حفظ العمل الذي أنجزته حتى هذه اللحظة، حتى لا تضطر إلى إعادة العمل مرة ثانية، إذا انهار الحاسب لديك لسبب ما. لهذا اتبع الخطوات التالية لحفظ العمل:

- اختر **Save Project** من القائمة **File**.

يستجيب فيجول بيسك بحفظ كل التغيرات المنجزة على ملف المشروع أو أي من الملفات التابعة للمشروع (مثال ذلك، الملف Hello.frm).

إضافة الزر خروج إلى النموذج frmHello

حسب ما يظهر من الشكل 1–1، فالنموذج المكتمل سيحوي ثلاثة أزرار أوامر داخله، وهي: **إظهار الترحيب** و **مسح** و **خروج**.

لوضع زر أمر ما ضمن النموذج، لا بد لك من تحديده أولاً من مربع الأدوات.

إطار مربع الأدوات Toolbox Window

يحوي إطار مربع الأدوات، رموز جميع الكائنات المتاحة لمشروعك الحالي. ومهمتك هي التقاط الكائن من مربع الأدوات، ووضعه على النموذج.

- أظهر إطار مربع الأدوات، باختيار **Toolbox** من القائمة **View** لفيجول بيسك.

يستجيب فيجول بيسك بإظهار مربع الأدوات (الشكل 1–14).



الشكل 1–14

إطار مربع الأدوات

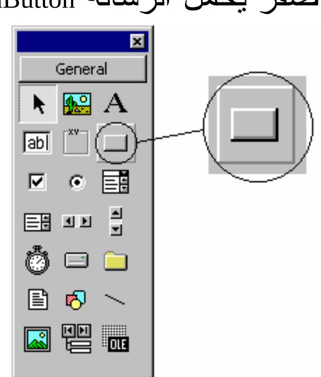
Toolbox.

ملاحظة

قد يختلف شكل إطار مربع الأدوات قليلاً عما هو عليه في الشكل 1-14، وذلك تبعاً للموقع الذي يأخذه على سطح مكتب فيجول بيسك (أي حسب المكان الذي تضعه فيه). كما أن إطار مربع الأدوات، قد يحوي رموزاً أكثر أو أقل، تبعاً لنوع نسخة فيجول بيسك 5 التي تملكها، وتبعاً لإعدادات فيجول بيسك 5.

يبين الشكل 1-15 رمز زر الأمر مكبراً، وهو طبعاً أحد الرموز التي تظهر في مربع الأدوات. تستطيع التعرف بسهولة على مختلف الرموز في مربع الأدوات بوضع مؤشر الفأرة فوق أي رمز بدون النقر عليه، ليظهر مستطيل أصفر يحمل بداخله اسم الكائن الذي يمثله هذا الرمز.

فمثلاً. عند وضع مؤشر الفأرة فوق رمز زر الأمر دون النقر عليه، يظهر مستطيل أصفر يحمل الرسالة CommandButton داخله.



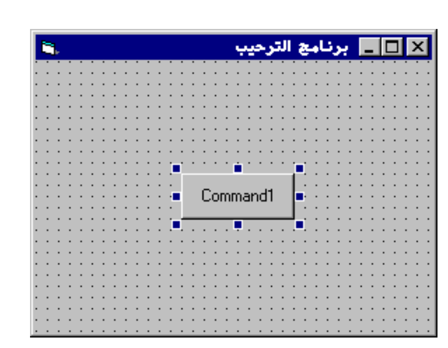
الشكل 1-15
رمز زر الأمر
CommandButton
في إطار مربع الأدوات.

وضع الزر خروج على النموذج

اتبع الخطوات التالية لوضع زر أمر على النموذج:

- انقر نقراً مزدوجاً على رمز زر الأمر في مربع الأدوات. (انظر الشكل 1-15 للتعرف على شكل زر الرمز).

يستجيب فيجول بيسك بوضع زر أمر في مركز النموذج (انظر الشكل 1-16).



الشكل 1-16 النموذج مع زر الأمر بداخله.

يتولى فيجول بيسك تعيين مختلف القيم الافتراضية لخصائص زر الأمر CommandButton الذي وضعته على النموذج. فمثلاً العنوان الافتراضي (Caption) لذلك الزر هو Command1.

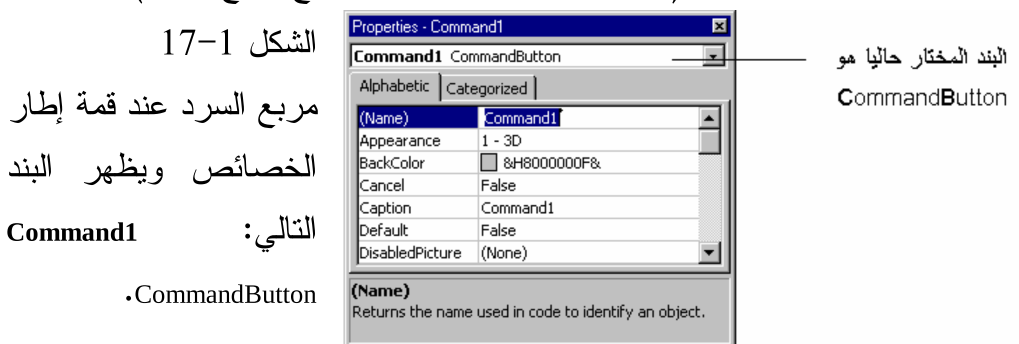
تغيير الخاصية Name للزر خروج (تغيير الاسم البرمجي)

ستغير اسم زر الأمر من Command1 إلى cmdExit باعتبار أن هذا الزر سيعمل زر الإنهاء **خروج**:

- اختر **Properties Window** من القائمة **View** لفيجول بيسك.

يستجيب فيجول بيسك بإظهار إطار الخصائص.

- تحقق أن مربع السرد عند قمة إطار الخصائص يظهر البند التالي: CommandButton **Command1**. (انظر الشكل 1-17 لتحديد موقع مربع السرد).



الشكل 1-17 مربع السرد عند قمة إطار الخصائص ويظهر البند التالي: **Command1** CommandButton.

ملاحظة

يمتلك النموذج الآن كائنين: النموذج frmHello وزر الأمر Command1. يُظهر إطار الخصائص، خصائص الكائن الذي يظهر اسمه حالياً في مربع السرد الواقع عند

قمة إطار الخصائص.
للانتقال بين الكائنات، انقر رمز السهم النازل، الموجود يمين مربع السرد واختر الكائن المطلوب من اللائحة المنسدلة للأسفل.

- غير الخاصية Name للزر Command1 إلى cmdExit.

لاحظ كيف وضعنا الحروف الثلاثة cmd قبل كلمة Exit، وذلك كما اتفقنا سابقاً، للدلالة على نوع الكائن ووظيفته في آن واحد (تعتبر وظيفة زر الأمر في مثالنا هذا، تنفيذ أمر الخروج من البرنامج)، هذا العمل ليس من متطلبات لغة فيجول بيسك، ولكن لتسهيل قراءة وفهم البرنامج، فعندما نشاهد الاسم cmdExit، نستطيع القول مباشرة، أن هذا الاسم يخص زر أمر وأن وظيفته هي إنهاء البرنامج.

تغيير الخاصية Caption للزر خروج (تغيير العنوان)

العنوان الافتراضي الذي يعطيه فيجول بيسك لزر الأمر هو Command1. وبما أن وظيفة زر الأمر هذا الخروج من البرنامج، فأنسب عنوان له هو **خروج:**

- غير الخاصية Caption لزر الأمر cmdExit من Command1 إلى &خروج.

للحصول على الرمز (&)، اضغط المفتاحين Shift+7 على لوحة المفاتيح. عند استخدام الرمز & (مثل استخدامه قبل الحرف **خ** في الخطوة السابقة)، يتسبب بقيام فيجول بيسك بوضع خط تحت الحرف الذي يليه (الحرف الذي يأتي بعد الرمز &)، ولاحظ وجود خط تحت الحرف **خ** في عنوان الزر **خروج**. انظر الشكل 1-1.
والآن، عند تنفيذ البرنامج، يؤدي الضغط على المفتاحين (خ+Alt) من لوحة المفاتيح إلى نفس تأثير النقر على الزر **خروج**.

ملاحظة

يُنصح دائماً باستخدام الرمز & قبل أحد حروف عنوان زر ما. يتسبب هذا الرمز & بظهور خط تحت الحرف الذي يليه مباشرة، وأثناء التنفيذ يصبح المستخدم قادراً، إما على نقر الزر بالفأرة أو ضغط المفتاح Alt إضافة لضغط الحرف المحدد (ضغط المفتاح Alt والمفتاح الذي تحته خط). والذي يمثل حرف وصول سريع.

تبديل موقع الزر خروج

كما ترى من الشكل 1–1، يجب أن يقع الزر **خروج** بقرب الحافة السفلى من النموذج. اسحب الزر **خروج** إلى الموقع المطلوب بضغط زر الفأرة الأيسر على أي مكان من الزر، ثم سحب وتحريك مؤشر الفأرة دون تحرير الزر الأيسر للفأرة، ثم تحرير زر الفأرة عند الوصول للموقع المناسب.

تبديل خصائص الخط (Font) للزر خروج

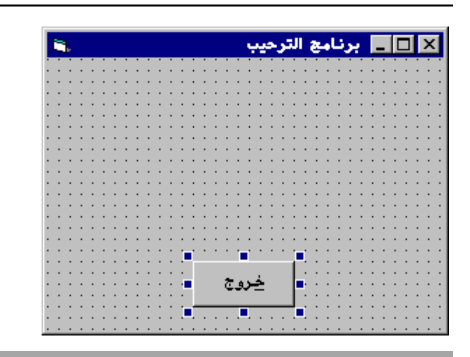
كما تلاحظ من الشكل 1–1، يختلف نوع الخط المستخدم في الزر **خروج** عن الخط الافتراضي الذي استخدمه فيجول بيسك لعنوان الزر، الذي وضعته على النموذج.
اتبع الخطوات التالية لتبديل نوع الخط للزر **خروج**:

- انقر على الخلية الواقعة يمين الخاصية Font التابعة للزر cmdExit. كما تلاحظ تحوي الخلية على مربع داخله ثلاث نقاط، يؤدي نقره إلى فتح مربع حوار تحديد الخطوط، يمكّنك من اختيار قيم معينة بواسطته.
- انقر على الزر ذى الثلاث نقط الموجود جانب الخاصية Font للزر cmdExit.

يستجيب فيجول بيسك بإظهار مربع الحوار Font.

- غير نوع الخط إلى System.
- غير حجم الخط إلى 10.
- انقر على الزر **Ok** في مربع الحوار Font.
- ألقِ نظرة على الزر **خروج** الآن، ولاحظ أن العنوان مكتوب بالخط الذي حددته مسبقاً بواسطة مربع الحوار Font.

يشبه النموذج الذي نبينه حتى الآن، ذاك المبين في الشكل 1–18.



الشكل 1−18
النموذج يحتوي الزر
خروج على سطحه.

ملاحظة

لعل إحدى المزايا الرئيسية لفيجول بيسك، أنه يمكّنك من رؤية نتائج البرمجة المرئية لحظياً. تمرّن دوماً وجرّب خيارات مختلفة (كأن تجرب أنواع خطوط مختلفة، وأحجام وألوان مختلفة) حتى تقنع بالنتائج.

احفظ العمل الذي أنجزته:

- اختر **Save Project** من القائمة **File** لفيجول بيسك.

إضافة أزرار أخرى إلى النموذج frmHello

لعل الوقت قد حان، لإضافة زرين جديدين إلى النموذج؛ الزر **إظهار الترحيب** والزر **مسح**.

وضع الزرين على النموذج

يحوي النموذج حسب ما يفترضه الشكل 1−1، زري أمر آخرين:

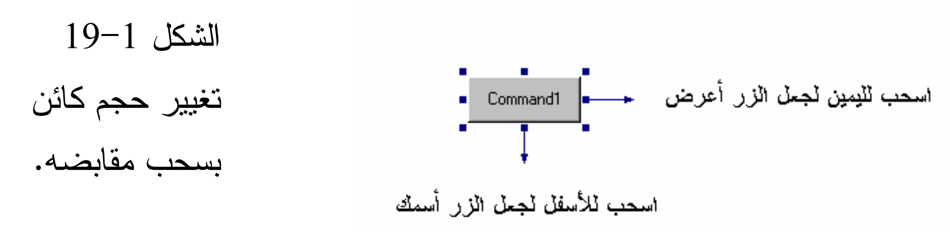
الزر **إظهار الترحيب** والزر **مسح**، سنضع هذين الزرين على النموذج:

- أضف الزر **إظهار الترحيب** إلى النموذج بالنقر المزدوج على رمز زر الأمر Command Button ضمن مربع الأدوات. اسحب زر الأمر الجديد إلى اليمين (سيأخذ هذا الزر دور الزر **إظهار الترحيب**).
- انقر نقراً مزدوجاً رمز زر الأمر في مربع الأدوات مجدداً، ثم اسحب زر الأمر الجديد إلى اليسار. (يأخذ هذا الزر دور الزر **مسح**).

تغيير حجم الأزرار

الأحجام الافتراضية للزرين **إظهار الترحيب** و **مسح** أصغر مما هي عليه في الشكل 1-1.

□كبّر الزرين الجديدين اللذين وضعتهما للتو في النموذج، يتم تكبير أو تصغير كائن باختياره أولاً. يحيط فيجول بيسك الكائن حال اختياره بمستطيل يتألف من ثماني مربعات سوداء، تدعى بالمقابض. اسحب أحد المقابض حتى يصل الكائن إلى الحجم المطلوب، فمثلاً لتغيير حجم الكائن أفقياً اسحب أحد المقابض أفقياً، أما لتغيير حجم الكائن عمودياً فاسحب أحد مقابضه عمودياً (انظر الشكل 1-19).



الشكل 1-19
تغيير حجم كائن بسحب مقابضه.

تبديل خاصية الاسم Name للزرين السابقين

الاسمان الافتراضيان الذين يطلقهما فيجول بيسك على الزرين الذين وضعتهما للتو في النموذج هي Command1 و Command2، ولكننا سنستبدلهما باسمين يناسبان عملهما أكثر:

□بدّل الخاصية Name لزر الأمر اليميني إلى cmdHello.

□بدّل الخاصية Name لزر الأمر اليساري إلى cmdClear.

تبديل عنواني الزرين السابقين

حسب ما يوضحه الشكل 1-1، يجب أن يكون عنوان الزر الأيمن هو **إظهار الترحيب**، وأن يكون عنوان الزر الأيسر هو **مسح**.

□بدّل الخاصية Caption للزر الأيمن إلى إ&ظهار الترحيب.

□بدّل الخاصية Caption للزر الأيسر إلى م&سح.

تبديل نوع الخط المستخدم في الزرين

- بدّل الخاصية Font للزر cmdHello إلى System واجعل حجم الخط مساوياً إلى 10.
- بدّل الخاصية Font للزر cmdClear إلى System واجعل حجم الخط مساوياً إلى 10.

ستكتشف بعد هاتين الخطوتين أن مساحة الزر صغيرة جداً، لهذا كّبر مساحة الزر، بسحب مقابضه.

يفترض أن يبدو النموذج لدى اكتماله كما في الشكل 1−20.



الشكل 1−20
النموذج بعد انتهاء تصميمه
مع ثلاثة أزرار داخله.

احفظ العمل المنجز حتى الآن:

- اختر **Save Project** من القائمة **File** لفيجول بيسك.

إضافة كائن مربع نص (Text Box) إلى النموذج frmHello

هناك كائن آخر يجب إضافته إلى النموذج، وهو كائن مربع النص.

مربع النص عبارة عن مساحة مستطيلة، يتم إظهار نصوص أو كتابة نصوص فيها.

يسمى مربع النص أحياناً بمربع تحرير.

وضع مربع النص في النموذج

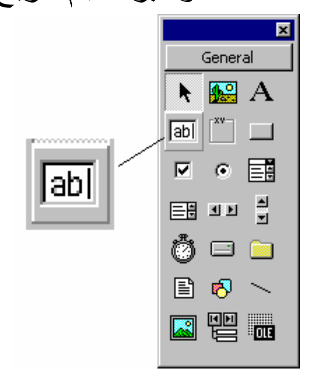
يبين الشكل 1−21 موقع رمز مربع النص Text Box في مربع الأدوات، وطبعاً قد تختلف مواقع الرموز في إطار مربع الأدوات عما هي عليه في الشكل 1−21 تبعاً للإصدار المستخدم للغة ولإعداداتها.

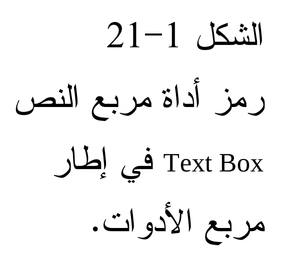
ملاحظة

يظهر مستطيل أصفر يحوي النص TextBox بداخله، عند وضع مؤشر الفأرة فوق رمز مربع النص (دون ضغط أزرار الفأرة).

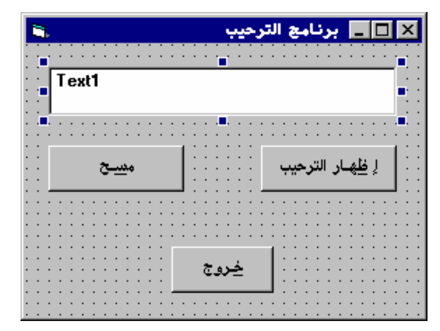
اتبع الخطوتين التاليتين لوضع مربع نص Text Box على النموذج:

- انقر نقراً مزدوجاً على رمز مربع النص Text Box ضمن إطار مربع الأدوات.
- انقل وغير حجم مربع النص حتى يظهر كما في الشكل 1-22.





الشكل 1-21
رمز أداة مربع النص
Text Box في إطار
مربع الأدوات.



الشكل 1-22
النموذج بعد وضع
مربع نص بداخله.

تبديل خصائص مربع النص

اتبع الخطوات التالية لتعديل بعض خصائص مربع النص:

- غيّر الخاصية Name لمربع النص من الاسم الافتراضي Text1 إلى .txtDisplay

خاصية Text الافتراضية لمربع النص هي Text1، وبالتالي فعند تنفيذ برنامج الترحيب يظهر النص Text1 ضمن مربع النص. نرغب أن يكون هذا المربع خالياً عند بدء

تشغيل البرنامج، لهذا، احذف النص الذي يظهر في الخلية الواقعة يمين الخاصية Text للكائن txtDisplay.

- غيّر الخاصية Font لمربع النص txtDisplay إلى System واجعل حجم الخط مساوياً إلى 10.

تساوي القيمة الافتراضية للخاصية Alignment لمربع النص إلى 0-LeftJustify مما يعني أن النص في مربع النص يبدأ من اليسار. غيّر هذه الخاصية إلى 2-Center لأننا نريد للنص الظهور في وسط مربع النص.

يرفض فيجول بيسك وضع النص في مركز مربع النص Text Box، ما لم يتم إسناد القيمة True إلى الخاصية MultiLine لمربع النص. لهذا يجب تغيير الخاصية MultiLine إلى True إلى جانب تغيير الخاصية Alignment إلى 2-Center. (يؤدي إسناد القيمة True إلى الخاصية MultiLine إلى تمكين فيجول بيسك من إظهار أكثر من سطر واحد من مربع النص). إذاً بدّل الخاصية MultiLine لمربع النص txtDisplay إلى True.

احفظ العمل:

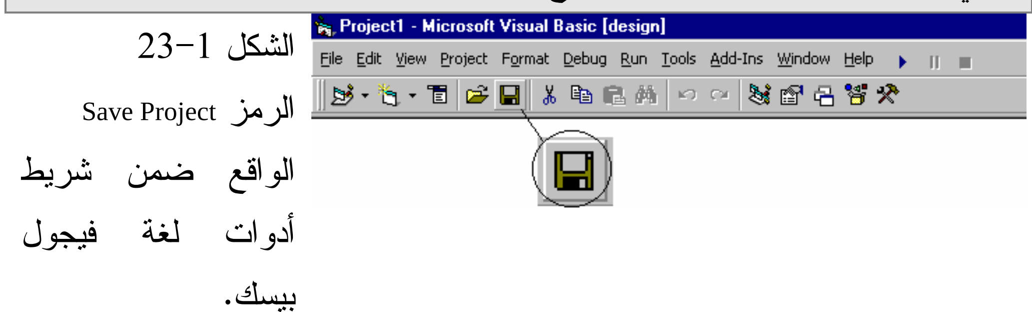
- اختر **Save Project** من القائمة **File** لفيجول بيسك.

ملاحظة

حسب ما ذكرنا سابقاً، من الهام جداً حفظ العمل المنجز من وقت لآخر. طريقة أولى لحفظ العمل تتمثل باختيار **Save Project** من القائمة **File** لفيجول بيسك. لكن هنالك طريقة أسهل بكثير لحفظ العمل. ألقِ نظرة على الشكل 1−23، يظهر هذا الشكل شريط أدوات لغة فيجول بيسك. (إذا لم يكن شريط الأدوات ظاهراً، اذهب إلى القائمة **View** واختر Toolbars ثم انقر على العنصر Standard الذي يظهر في القائمة المنبثقة). نعود إلى الشكل 1−23. يوجد على شريط الأدوات رمز يظهر كقرص مرن، إذا وضعت مؤشر الفأرة (دون النقر على أحد أزرارها) فوق هذا الرمز، يظهر مستطيل أصفر يحمل الرسالة Save Project من القائمة **File**. أجل، لقد سهل مصممو فيجول بيسك كثيراً عملية حفظ المشروع، فنقرة واحدة من وقت لآخر تضمن

لك السلامة.

هناك طريقة سريعة أخرى لحفظ آخر التعديلات التي أجريتها على النموذج النشط الحالي، اضغط فقط المفتاحين Ctrl+S سوياً، ليحفظ فيجول بيسك تعديلات النموذج الحالي فقط، وليس كامل ملفات المشروع.



الشكل 1−23 الرمز Save Project الواقع ضمن شريط أدوات لغة فيجول بيسك.

عملية بناء النماذج انطلاقاً من الرسوم التوضيحية والجداول

انتهى جزء التصميم المرئي الآن.

سنطالبك في كتابنا هذا ببناء عدد هائل من النماذج. لكن لن نستمر بهذه الطريقة في بناء النماذج، أي لن نبني النموذج خطوة بخطوة، بل سنــزودك بشكل النموذج المكتمل (كما في الشكل 1−1)، كما سنعطيك جدولاً يدعى جدول خصائص النموذج. يحوي جدول خصائص النموذج كل الكائنات المحتواة في النموذج ويستعرض كل الخصائص التي تختلف عن الخصائص الافتراضية لهذه الكائنات.

عملك هو اللحاق بالجدول، سطراً سطراً وتغيير قيم الخصائص إلى القيم التي تظهر في الجدول. الجدول 1−1 هو جدول خصائص النموذج frmHello الذي أنهينا بناءه للتو.

الجدول 1−1. جدول خصائص النموذج frmHello.

الكائن	الخاصية	القيمة
Form	**Name**	**FrmHello**
	BackColor	Blue
	Caption	برنامج الترحيب
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	FontName	System
	FontSize	10
	RightToLeft	True
CommandButton	**Name**	**cmdClear**
	Caption	&مسح
	FontName	System
	FontSize	10
	RightToLeft	True
CommandButton	**Name**	**cmdHello**
	Caption	إ&ظهار الترحيب
	FontName	System
	FontSize	10
	RightToLeft	True
TextBox	**Name**	**txtDisplay**
	Alignment	2-Center
	FontName	System
	FontSize	10
	MultiLine	True
	RightToLeft	True

ربط الكائنات بنصوص برمجية

باعتبار أننا وضعنا الكائنات في النموذج وحددنا خصائصها، فقد انتهى جزء التصميم المرئي (البرمجة المرئية Visual Programming).

الآن، حان الوقت لكتابة النصوص البرمجية لهذه الكائنات.

تعتبر لغة فيجول بيسك لغة مقادة بالأحداث event-driven programming language. هذا يعني أن نص البرنامج ينفذ استجابة لحادثة ما. فمثلاً يؤدي نقر الزر **خروج** أثناء تنفيذ برنامج الترحيب إلى توليد الحادثة Click آلياً، وبالتالي ينفذ نص البرنامج المرافق لحادثة نقر الزر **خروج** آلياً أيضاً.

كذلك، تتولد الحادثة Click (حادثة النقر) أيضاً عند نقر الزر **إظهار الترحيب**، وينفذ نص البرنامج المرافق لحادثة نقر هذا الزر آلياً.

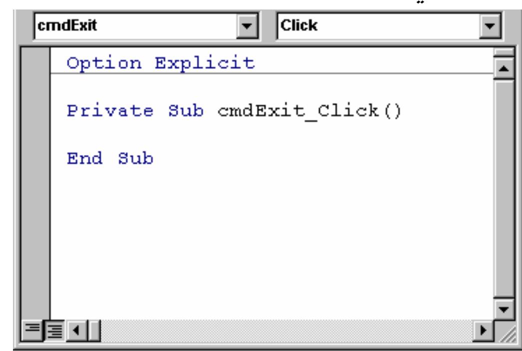
عملك هو كتابة النص المناسب وربطه بالكائن والحادثة. هل يبدو هذا الأمر معقداً؟ بالواقع إنه سهل جداً!، إذاً لنبدأ بربط بعض النص البرمجي بالحادثة Click للزر **خروج**.

ربط الزر خروج بنص برمجي

اتبع الخطوات التالية لربط نص برمجي بالزر cmdExit:

- انقر نقراً مزدوجاً على الزر cmdExit.

يستجيب فيجول بيسك بإظهار إطار نص البرنامج (سنكتب برنامجاً في هذا الإطار) المبين في الشكل 1-24.



الشكل 1-24

ربط الزر **خروج** بنص البرنامج المرافق، في إطار نص البرنامج Code Window.

يُسهّل فيجول بيسك إلى حد كبير التعرف على النص الظاهر حالياً في إطار نص البرنامج.

فكما يتبين من الشكل 1–24، يُظهر مربع السرد اليساري اسم الكائن cmdExit، ويُظهر مربع السرد اليميني اسم الحادثة Click.

كما يظهر من الشكل 1–24 أيضاً، فقد أضاف فيجول بيسك مسبقاً سطري نص يمثلان بداية ونهاية الإجراء الخاص بالكائن المحدد والحادثة المختارة، في إطار نص البرنامج:

```
Private Sub cmdExit_Click()

End Sub
```

سنكتب الآن نص البرنامج ضمن هذين السطرين.

السطران الأول والأخير من نص البرنامج

السطر الأول من النص (الذي كتبه عنك فيجول بيسك) يبدأ بالكلمتين Private Sub.

الكلمة Sub عبارة عن كلمة محجوزة في لغة فيجول بيسك، تدل على أن الإجراء Procedure يبدأ من هنا.

والإجراء Procedure عبارة عن نص برنامج مكرس لحادثة خاصة. اسم الإجراء في مثالنا هو cmdExit_Click().

السطر الأخير من النص، كُتب أيضاً من قبل لغة فيجول بيسك. وتشير العبارة End Sub لنهاية الإجراء.

اسم الإجراء

اسم الإجراء هو cmdExit_Click(). ما السبب الذي دفع فيجول بيسك إلى تعيين اسم الإجراء بنفسه؟. تسبب النقر المزدوج على الزر cmdExit بمعرفة فيجول بيسك أنك تحاول ربط ذلك الزر بنص برمجي، وبذلك حدد القسم الأول من اسم الإجراء وهو cmdExit.

يمثل القسم الثاني من اسم الإجراء نوع الحادثة الناجمة (الحادثة Click).

أيضاً ما الذي جعل فيجول بيسك يحدد هذا الاسم (Click) ؟!.

السبب طبعاً أنها تُمثل الحادثة التي ترغب بكتابة برنامج لها.

لاحظ أن القسمين الأول والثاني من اسم الإجراء يُفصلان عن بعضهما برمز الخط التحتي (_)، وبذلك يصبح اسم الإجراء كالتالي: cmdExit_Click().

ولاحظ أيضاً أن آخر رمزين في اسم الإجراء هما القوسين ().

نص برنامج الإجراء cmdExit_Click()

ما هو نص البرنامج الذي يتوجب عليك كتابته في الإجراء cmdExit_Click()؟. باعتبار أن هذا الإجراء يُنفذ عند نقر الزر **خروج** (إنهاء البرنامج)، فالنص الواجب عليك كتابته ضمن هذا الإجراء يجب أن يتسبب بإنهاء البرنامج. تُستخدم العبارة End لإنهاء البرنامج، وبالتالي فنص البرنامج اللازم كتابته ما هو إلا العبارة End فحسب.

- اكتب End في الإجراء cmdExit_Click(). يُفترض أن يبدو الإجراء بعد كتابة هذه العبارة بالشكل التالي:

```
Private Sub cmdExit_Click()
    End
End Sub
```

هذا كل ما في الأمر، لقد انتهيت للتو من ربط الإجراء cmdExit_Click() بنص البرنامج المناسب.

تنفيذ برنامج الترحيب

رغم أنك لم تنته بعد من ربط باقي الكائنات بنصوصها البرمجية، لكن هذا لا يمنع من تنفيذ برنامج الترحيب ورؤية كيف يعمل نص البرنامج الذي ربطناه بالزر **خروج**.

- احفظ المشروع باختيار **Save Project** من القائمة **File**، أو انقر رمز **Save Project** (رمز القرص المرن) على شريط أدوات فيجول بيسك.
- اختر **Start** من القائمة **Run** لفيجول بيسك.

يستجيب فيجول بيسك بتنفيذ البرنامج. يظهر إطار هذا البرنامج بشكل يماثل ذلك المبين في الشكل 1-1.

تستطيع النقر على الزر **إظهار الترحيب** أو على الزر **مسح**، لكن لن يحدث شيء، والسبب طبعاً أننا لم نربط بعد هذين الزرين بأي نص برمجي.

▫انقر الآن الزر **خروج**، فينفذ الإجراء cmdExit_Click() استجابة لحادثة النقر، وباعتبار أن نص هذا الإجراء يحوي العبارة End، فسيؤدي تنفيذه إلى إنهاء عمل برنامج الترحيب.

إضافة المزيد إلى نص الإجراء cmdExit_Click()

اتبع الخطوات التالية لربط الإجراء cmdExit_Click() بالمزيد من النص البرمجي:

▫انقر نقراً مزدوجاً على الزر cmdExit_Click().

يستجيب فيجول بيسك بإظهار cmdExit_Click() جاهزاً للتعديل من قبلك.

▫أضف العبارة Beep قبل العبارة End كما يلي:

```
Private Sub cmdExit_Click()
    Beep
    End
End Sub
```

▫احفظ العمل المنجز.

▫نفّذ برنامج الترحيب باختيار **Start** من القائمة **Run** لفيجول بيسك.

تتسبب العبارة Beep بإصدار رنين (صافرة) من قبل الحاسب الشخصي. وبالتالي يؤدي نقر الزر **خروج** إلى إصدار صوت صافرة، ثم إنهاء برنامج الترحيب.

▫انقر الزر **خروج** وتحقق من أن البرنامج أصدر رنيناً، ثم أنهى نفسه.

ربط الزر إظهار الترحيب بنص البرنامج المناسب

اتبع الخطوات التالية لربط الزر **إظهار الترحيب** بنص برنامج:

▫أظهر النموذج، ثم انقر نقراً مزدوجاً على الزر **إظهار الترحيب**، (افترضنا أنك ستُظهر النموذج باختيار **Project Explorer** من القائمة **View**، ثم نقر البند

frmHello في إطار المشروع Project، ثم نقر الرمز **View Object** الذي يظهر كثاني رمز على اليسار عند قمة إطار المشروع).

*يستجيب فيجول بيسك بعد النقر المزدوج على الزر **إظهار الترحيب** بإظهار الإجراء cmdHelloDisplay_Click() مع سطري البرنامج التاليين:*

```
Private Sub cmdHelloDisplay_Click()
End Sub
```

يُنفذ الإجراء السابق عند نقر الزر **إظهار الترحيب** أثناء تنفيذ برنامج الترحيب.

يا ترى ما هو نص البرنامج الواجب ربطه بهذا الإجراء؟ يعتمد هذا على ما سيحصل عند نقر الزر **إظهار الترحيب**.

ففي مثالنا هذا يُطلب من برنامج الترحيب إظهار رسالة الترحيب **مرحباً بكم** في مربع النص.

- أدخل النص التالي ضمن الإجراء cmdExit_Click():

```
    txtDisplay.Text = "مرحباً بكم"
```

يفترض أن يبدو الإجراء كما يلي عند الانتهاء:

```
Private Sub cmdHelloDisplay_Click()
    txtDisplay.Text = "مرحباً بكم"
End Sub
```

يمثل txtDisplay اسم الكائن Text Box (مربع النص الذي سيُظهر الكلمتين **مرحباً بكم** داخله).

تسند العبارة التالية:

```
    txtDisplay.Text = "مرحباً بكم"
```

القيمة **مرحباً بكم** إلى الخاصية Text لمربع النص txtDisplay. (تمثل قيمة الخاصية Text النص الذي سيظهر في مربع النص txtDisplay).

ملاحظة

يُمكن استخدام الصيغة التالية لإسناد قيمة جديدة إلى خاصية ما، ضمن نص برنامج:

```
ObjectName.Property = قيمة جديدة
```

فمثلاً، لتبديل الخاصية Text لمربع النص txtDisplay إلى القيمة **مرحباً بكم** استخدم

العبارة التالية:

```
txtDisplay.Text = "مرحباً بكم"
```

انتبه لكتابة النقطة (.) بين اسم الكائن txtDisplay واسم الخاصية Text وبدون فراغات بينهما.

ربط الزر مسح بنص برنامج مناسب

اتبع الخطوات التالية لربط الزر **مسح** بنص برنامج الحادثة Click:

- أظهر النموذج، وانقر نقراً مزدوجاً على الزر **مسح**.

*يستجيب فيجول بيسك للنقر المزدوج على الزر **مسح**، بإظهار إطار نص البرنامج Code Window عند الإجراء cmdClear_Click() وجاهزاً للتعديل.*

يفترض أن يقوم هذا الإجراء بمسح محتويات مربع النص. أي بمعنى آخر يجب تحويل قيمة الخاصية Text لمربع النص، إلى سلسلة صفرية (فراغ "")، يمكن إنجاز ذلك بإضافة العبارة التالية إلى الإجراء cmdClear_Click():

```
txtDisplay.Text = ""
```

- اكتب العبارة التالية في الإجراء cmdClear_Click():

```
txtDisplay.Text = ""
```

يُفترض أن يبدو الإجراء لدى اكتماله كما يلي:

```
Private Sub cmdClear_Click()
    txtDisplay.Text = ""
End  Sub
```

- احفظ العمل المنجز باختيار **Save Project** من القائمة **File** لفيجول بيسك (أو انقر على رمز القرص المرن على شريط أدوات لغة فيجول بيسك).

تنفيذ برنامج الترحيب مرة ثانية

اكتمل برنامج الترحيب الآن. اتبع الخطوات التالية لتنفيذه:

- اختر **Start** من القائمة **Run** لفيجول بيسك أو اضغط المفتاح F5 لبدء البرنامج.

ملاحظة

إحدى الأسباب الرئيسية التي تجعل فيجول بيسك بهذا الشيوع حقيقة، أنك تستطيع تطوير برنامجك قليلاً ثم تنفيذه ورؤية نتائج عملية التطوير على أرض الواقع، ثم متابعة تطوير البرنامج بعض الشيء وتشغيل التطبيق لاختيار عملية التطوير . . . وهكذا دواليك.

ذكرنا، أنك تستطيع تنفيذ التطبيق المكتوب بلغة فيجول بيسك باختيار **Start** من القائمة **Run** لفيجول بيسك.

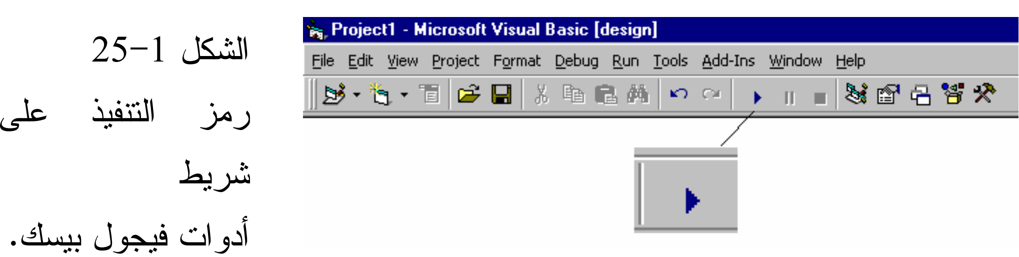
كما تلاحظ، تتطلب عملية تطوير التطبيق تكرار تنفيذه كثيراً، ولذلك راعى مصممو اللغة هذا الجانب، فوضعوا رمز التنفيذ على شريط أدوات لغة فيجول بيسك ويدعى هذا الرمز **Start** (انظر الشكل 1−25). ويكفي نقر الرمز **Start** على شريط الأدوات لتنفيذ التطبيق الراهن.

إذا كان شريط الأدوات غير ظاهر على سطح مكتب فيجول بيسك لسبب ما، اختر **Toolbars** من القائمة **View** لفيجول بيسك ثم اختر **Standard** من القائمة التي تظهر.

إذاً، تعتبر ميزة تنفيذ البرامج أثناء عملية التطوير، أمراً بالغ الأهمية، ففي لغات البرمجة الأخرى مثل Visual C++، يتوجب أولاً ترجمة ثم ربط البرنامج (Compile ثم Link) قبل التمكن من تنفيذه. وهذه العملية تأخذ وقتاً طويلاً نوعاً ما. يُسهل فيجول بيسك عملية تنفيذ التطبيق الذي يجري تطويره إلى حد بعيد، لكن قد يظهر خطأ ما في البرنامج ويتسبب بانهيار النظام كاملاً، مما يُجبرك على إعادة تشغيل Windows أو فيجول بيسك مرة أخرى. يعني هذا أن عدم حفظ العمل قد يتسبب بضياعه وسنضطر إلى بدء عملية التطوير من جديد.

لهذا كن حكيماً، واجعل عملية حفظ العمل قبل تنفيذه قانوناً تسير عليه. والأمر جداً بسيط، فكما ذكرنا يقع رمز حفظ المشروع **Save Project** على شريط الأدوات. ويكفي نقره لحفظ العمل عند المرحلة الراهنة. لهذا انقر أولاً على الرمز

Save Project ثم انقر بعد ذلك الرمز **Start** (انظر الشكل 1−25) لتنفيذ التطبيق. وبهذه الطريقة، فإن وقوع خطأ ما أثناء تنفيذ البرنامج، لن يتسبب بضياع المشروع أو العمل، وتستطيع إعادة تشغيل فيجول بيسك مجدداً، وفتح المشروع المخزن وتصحيح الخطأ ومعاودة تنفيذ التطبيق.



الشكل 1−25 رمز التنفيذ على شريط أدوات فيجول بيسك.

- انقر الزر **إظهار الترحيب** وأيضاً الزر **مسح** لإظهار ثم مسح الرسالة مرحباً بكم، في مربع النص. يجب أن يظهر إطار برنامج الترحيب كما في الشكل 1−2، بعد
- نقر الزر **إظهار الترحيب**. بينما سيظهر كما في الشكل 1−1 عند نقر الزر **مسح**.
- تستطيع استخدام المفاتيح Alt+ظ و Alt+س للحصول على نفس الاستجابة عند نقر الزر **إظهار الترحيب** وعند نقر الزر **مسح**. (طبعاً، بسبب وجود خط تحت الحرف **ظ** في عنوان الزر **إظهار الترحيب**، وخط تحت الحرف **س** في عنوان الزر **مسح**).
- يمكن إنهاء البرنامج بنقر الزر **خروج**، أو بضغط المفتاحين Alt+خ على لوحة المفاتيح.

أحداث أخرى

يستخدم برنامج الترحيب الحادثة Click لأزرار الأوامر، (فمثلاً، تقع الحادثة Click لزر الأمر cmdExit عند نقر الزر **خروج** مما يتسبب بالتنفيذ الآلي للإجراء cmdExit_Click(). هنالك حوادث أخرى يمكن استخدامها في البرنامج. وكل حادثة تمتلك بدورها إجراءها الخاص.

الحادثة KeyDown

سنتعرف على الإجراء الذي يرافق الحادثة KeyDown، والتي تحصل عند ضغط مفتاح ما على لوحة المفاتيح.

- اتبع الخطوات التالية لرؤية الإجراء KeyDown للزر **خروج**.
- انقر نقراً مزدوجاً على الزر **خروج**.

يظهر فيجول بيسك في الحالة الافتراضية الإجراء المرافق للحادثة Click.

- باعتبار أننا لا نريد تعديل الإجراء المرافق للحادثة Click وإنما نريد تعديل إجراء الحادثة KeyDown، لهذا انقر على مربع السرد الذي يظهر في الجانب الأيمن عند قمة إطار النص.

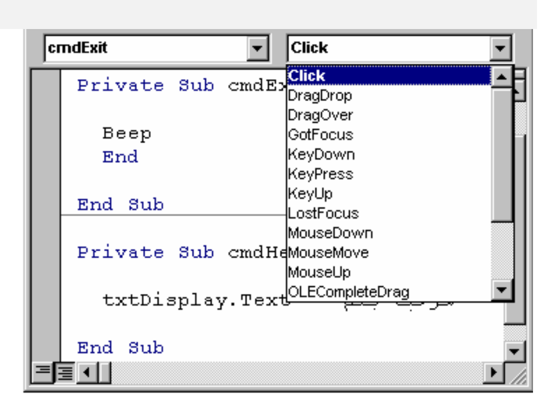
يستجيب فيجول بيسك بسرد لائحة تحوي كل الحوادث المتاحة للكائن المدعو cmdExit (انظر الشكل 1-26).

- اختر البند KeyDown من اللائحة.

يستجيب فيجول بيسك بإظهار الإجراء cmdExit_KeyDown():

```
Private Sub cmdExit_KeyDown(KeyCode AS Integer,Shift As Integer)

End Sub
```



الشكل 1-26

إظهار الحوادث المتاحة

للزر cmdExit.

لاحظ أن السطر الأول من الإجراء والذي يكتبه فيجول بيسك آلياً، يختلف قليلاً عن السطر الأول للحادثة Click فقوسي الإجراء cmdExit_Click()، لا يحويان شيئاً داخلهما. أما قوسي الإجراء cmdExit_ KeyDown() فيحويان قدراً من النص البرمجي. سنتعرف خلال الفصول القادمة على هذا الاختلاف بشكل أوضح.

أما عند هذه النقطة، فلا تُضف شيئاً إلى الإجراء cmdExit_ KeyDown() (مضينا في هذا المثال، لمجرد إطلاعك على أشكال أخرى من الحوادث التي ترافق كائن ما في فيجول بيسك).

إنشاء الملف التنفيذي للمشروع (Hello.exe)

نفذنا سابقاً برنامج الترحيب باختيار **Start** من القائمة **Run**. طبعاً يختلف الأمر عند الانتقال إلى الواقع العملي، فلا أحد يرضى أن يُشغل التطبيق الذي بحوزته بهذه الطريقة، بل قد لا يملك المستخدم الذي اقتنى التطبيق، لغة فيجول بيسك أصلاً.

يجب أن تكون قادراً إذاً على تحويل البرنامج إلى ملف تنفيذي، حتى نتمكن من توزيع التطبيق الذي طورناه.

فمثلاً يجب تحويل برنامج الترحيب إلى ملف تنفيذي Hello.exe:

- اختر Make Hello.exe File من القائمة **File** لفيجول بيسك.

*يستجيب فيجول بيسك بإظهار مربع الحوار Make. (لا تنقر الزر **OK** في مربع الحوار في هذه اللحظة).*

- استخدم مربع الحوار لحفظ الملف Hello.exe في الدليل C:\VB5Prg\Ch01.
- انقر الزر **OK** الآن.

يستجيب فيجول بيسك بحفظ الملف Hello.EXE في الدليل C:\VB5Prg\Ch01.

تستطيع الآن تنفيذ Hello.exe كأي برنامج آخر في Windows!، تستطيع مثلاً استخدام مستكشف Windows ثم النقر المزدوج على الملف Hello.exe.

ملف تنفيذي EXE صغير الحجم

ألقِ نظرة على الملف Hello.exe الموجود حالياً ضمن الدليل C:\VB5Prg\Ch01. كيف يتمكن ملف بحجم 8 كيلو بايت فقط من إنجاز كل الأشياء التي يستطيع Hello.exe إنجازها؟.

فكر بالأمر، ينجز البرنامج Hello.exe أموراً كثيرة، فيمكنك نقر الأزرار الثلاثة، ويسمح مثلاً بسحب إطار البرنامج، يمتلك البرنامج أيضاً مربع نص داخله، بمعنى

آخر، يمتلك هذا البرنامج الصغير كل معالم البرامج القياسية في Windows. والسبب وراء صغر حجم الملف التنفيذي Hello.exe، أن البرنامج يفترض وجود الملف Msvbvm50.DLL في الدليل System. والدليل System موجود طبعاً ضمن دليل Windows.

فمثلاً ترى الدليل C:\Windows\System إذا كان نظام التشغيل المستخدم هو Windows95 أو في الدليل C:\WinNT\System32 عندما يكون نظام التشغيل المستخدم هو Windows NT. المهم بغض النظر عن نظام التشغيل المستخدم يحتاج البرنامج Hello.exe أن يكون الملف Msvbvm50.DLL موجوداً ضمن دليل النظام System حتى يعمل. تستطيع العودة إلى ذلك الدليل والتأكد من وجود الملف المذكور. لقد تم تثبيت هذا الملف عند تثبيت فيجول بيسك 5 على قرصك الصلب.

حجم الملف Msvbvm50.DLL يساوي 1.3 ميجا بايت تقريباً، طبعاً يعتبر كبيراً، لكن حال توفر هذا الملف يصبح بوسع المستخدمين تنفيذ برامج بالغة القوة كتبت بلغة فيجول بيسك 5.

النبأ السار أن هذه البرامج القوية صغيرة الحجم جداً. تستطيع أيضاً رؤية الملف Hello.vbw في الدليل الذي حفظنا فيه الملف Hello.exe. الملف Hello.vbw يستخدم من قبل فيجول بيسك، ولا يتوجب عليك تزويد المستخدمين بهذا الملف.

ملاحظة

يتوجب عليك تزويد البرامج التي توزعها بملفات DLL اللازمة لها. فمثلاً يجب أن يكون الملف Msvbvm50.DLL موجوداً ضمن الدليل System على الحاسب الذي ستنفذ فيه البرامج المطورة بلغة V.B.5.

الخلاصة

كتبنا في هذا الفصل أول برنامج لك بلغة Visual Basic فتعلمت الخطوتين اللازمتين لكتابة البرنامج، وهما خطوة البرمجة المرئية Visual Programming Step وخطوة كتابة نص البرنامج Code Programming Step. يضع المبرمج الكائنات Objects في النموذج ويسند خصائصها بالقيم المناسبة أثناء خطوة البرمجة المرئية.

أما فى خطة كتابة النص Code Programming، فيتم انتقال الإجراء المرافق لكائن ما باختيار الكائن والحادثة ثم يكتب نص البرنامج المناسب فى الإجراء. ينفذ جزء البرنامج المكتوب هذا زمن التنفيذ عند وقوع الحادثة.

الفصل الثاني

الخصائص وعناصر التحكم والكائنات

يركز هذا الفصل على عناصر تحكم لغة فيجول بيسك، مثل شريطي التمرير الأفقي والعمودي Scroll Bar، ومربعات النص Text Boxes، وأزرار الخيارات Option Button، وأزرار الأوامر Command Button. ستتعلم من هذا الفصل كيف تضع هذه العناصر داخل برامجك، وكيفية تغيير خصائصها، وكيفية ربط النصوص البرمجية بها، (أي ربطها ببرامج متعلقة بها).

تُقدم معظم البرامج، معلومات إلى المستخدم، وتتلقى منه معلومات أيضاً. تُدعى عملية تبادل المعلومات بين التطبيق وبين المستخدم، بواجهة المستخدم User Interface. تَستخدم جميع برامج الويندوز، عناصر التحكم Controls، لتزويد المستخدم بواجهة سهلة ومفهومة (هذا من أهم أسباب شيوع النظام ويندوز). يوضح هذا الفصل مدى سهولة بناء واجهة استخدام جذابة في لغة فيجول بيسك.

عنصر تحكم شريط التمرير

يستخدم شريط التمرير Scroll Bar بكثرة في برامج ويندوز. يمكّنك استخدام هذا العنصر، من اختيار قيمة معينة، بوضع مؤشّر شريط التمرير عند موقع محدد منه، بدلاً من كتابة القيمة.

ملاحظة

أطلقنا على شريط التمرير في الفصل الأول كلمة كائن Object، لكن اعتباراً من هذا الفصل سنشير إليه بمصطلح عنصر تحكم Control.

في معظم الأحوال، يعتبر الكائن هو نفسه عنصر تحكم، لكن ليس دائماً، فالنموذج Form هو كائن، لكنه ليس عنصر تحكم. نستطيع استخدام كلمة كائن للدلالة على عنصر تحكم، إذا كان هذا العنصر سيوضع في نموذج.

برنامج السرعة

يوضح برنامج السرعة، كيفية استخدام شريط التمرير للحصول على قيمة معينة من المستخدم.

يُفترض في برنامج السرعة إنجاز ما يلي:

- يظهر الإطار المبين في الشكل 2-1 عند بدء تشغيل برنامج السرعة، يُفترض أن يوضع مؤشّر شريط التمرير عند مركز شريط التمرير (الموقع الافتراضي)، وأن تظهر الرسالة "50 كم / ساعة" (قيمة السرعة) ضمن مربع النص.



الشكل 2-1

نافذة برنامج السرعة.

■ينبغي على مربع النص، إظهار التغير في السرعة عند تغيير موضع مؤشّر شريط التمرير. فمثلاً يجب إظهار القيمة صفر عندما يوضع المؤشّر عند أقصى اليمين، أما عند وضعه عند أقصى اليسار فيجب إظهار القيمة 100.

□يؤدي نقر الزر **خروج** لإنهاء البرنامج.

التمثيل المرئي لبرنامج السرعة

يَستخدم برنامج السرعة عنصر تحكم شريط التمرير الأفقي. يبين الشكل 2-2 شكل عنصر التحكم هذا. طبعاً يختلف موضع هذا العنصر ضمن مربع الأدوات Toolbox تبعاً لاختلاف إصدار لغة فيجول بيسك المستخدم. ويؤدي وضع مؤشّر الفأرة فوق رمز شريط التمرير الأفقي، دون النقر عليه إلى ظهور مستطيل أصفر يحمل الرسالة HscrollBar داخله.

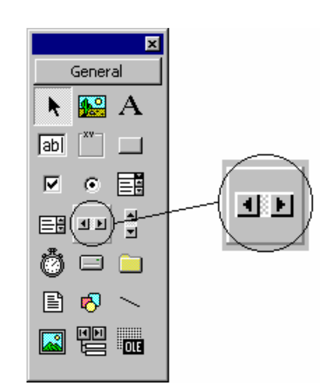
بهذه الطريقة، تتمكن من التحقق من وجود رمز شريط التمرير الأفقي في إطار مربع الأدوات.

□نفّذ فيجول بيسك، ثم انقر الزر **إلغاء الأمر** في الإطار New Project في حال ظهوره لإغلاق هذا الإطار، ثم اختر البند **New Project** من قائمة **File** للغة فيجول بيسك.

يستجيب فيجول بيسك بإظهار الإطار New Project.

□اختر الرمز Standard EXE من ضمن الإطار New Project، ثم انقر الزر **موافق**.

يستجيب فيجول بيسك بإنشاء مشروع جديد.



الشكل 2-2
رمز شريط التمرير الأفقي
داخل إطار مربع الأدوات.

سنحفظ الآن المشروع الجديد الذي أنشأناه:

□أنشئ الدليل C:\VB5Prg\Ch02.

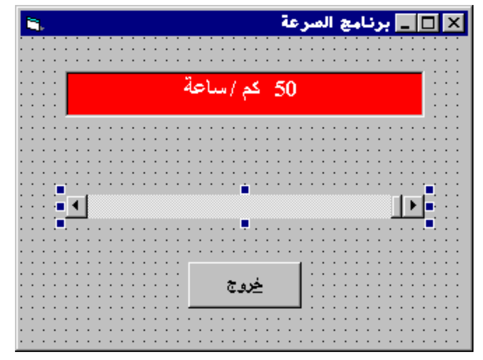
□تحقق من اختيار النموذج Form1، ثم اختر البند **Save Form1 As** من قائمة **File**.

يستجيب فيجول بيسك بإظهار مربع الحوار Save File As.

□استخدم مربع الحوار Save File As لحفظ النموذج باسم Speed.Frm في الدليل C:\VB5Prg\Ch02 واختر البند **Save Project As** من قائمة **File**، ثم استخدم مربع الحوار Save Project As لحفظ المشروع باسم Speed.vbp في الدليل نفسه.

□عدّل النموذج Form1 طبقاً للجدول 2-1.

يُفترض أن يبدو النموذج لدى اكتماله، كما في الشكل 2-3.



الشكل 2-3
النموذج frmSpeed
في طور التصميم.

الجدول 2−1. جدول خصائص النموذج frmSpeed.

الكائن	الخاصية	القيمة
Form	**Name**	**FrmSpeed**
	BackColor	Light gray
	Caption	برنامج السرعة
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
Horizontal Scroll Bar	**Name**	**hsbSpeed**
	Min	0
	Max	100
	RightToLeft	True
TextBox	**Name**	**txtSpeed**
	Alignment	2-Center
	Font	(اختر ما شئت)
	BackColor	Red
	ForeColor	White
	Text	50 كم / ساعة
	MultiLine	True
	RightToLeft	True

□ذكرنا في الفصل الأول، أن النقر المزدوج على رمز عنصر التحكم ضمن مربع الأدوات، يؤدي إلى وضع عنصر التحكم ذاك، ضمن النموذج الحالي.

يستجيب فيجول بيسك بوضع عنصر التحكم في وسط النموذج الحالي. تستطيع بعد ذلك نقله إلى مكان آخر عن طريق سحبه بمؤشّر الفأرة. كما تستطيع تكبيره أو تصغيره بسحب المقابض التي تظهر حوله.

□للولوج إلى خصائص عنصر التحكم، تأكد من اختيار هذا العنصر على النموذج

(أي توضّع المقابض حوله)، ثم اختر البند **Properties Windows** من قائمة **View**. أو تستطيع بدلاً من ذلك، النقر بالزر الأيمن للفأرة على العنصر، ثم اختيار البند **Properties** من القائمة الفرعية السريعة التي ظهرت.

يستجيب فيجول بيسك بإظهار إطار الخصائص Properties لعنصر التحكم المختار.

تستطيع الآن تغيير خصائص هذا العنصر.

- احفظ المشروع باختيار البند **Save Project** من قائمة **File** التابعة لفيجول بيسك.

كتابة نص برنامج السرعة

سنكتب الآن نص برنامج السرعة:

- اكتب النص التالي ضمن الإجراء ()cmdExit_Cilck التابع للنموذج frmSpeed:

```
Private Sub cmdExit_Click()
    End
End Sub
```

يُنفذ نص البرنامج السابق آلياً، عند نقر الزر **خروج**، ويُنهي تنفيذ برنامج السرعة.

ملاحظة

لإدخال نص البرنامج للزر **خروج**، انقر نقراً مزدوجاً على الزر **خروج** في مرحلة التصميم، فيظهر إطار البرنامج للإجراء ()cmdExit_Cilck. ويكون هذا الإجراء جاهزاً للتعديل من قبل المستخدم.

تأكد أن مربع السرد الواقع في الزاوية العليا اليسارية، من إطار البرنامج يحوي البند cmdExit، وأن المربع المجاور له يحوي البند Click.

- احفظ المشروع باختيار البند **Save Project** من قائمة **File** التابعة لفيجول بيسك.

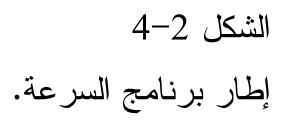
تنفيذ برنامج السرعة

لم ننته بعد من كتابة نصوص برنامج السرعة، لكن رغم ذلك، سننفذ برنامج السرعة لرؤية نتائج ما أنجزناه حتى الآن.

❑نفّذ برنامج السرعة (ضغط مفتاح F5 أو اختيار البند **Start** من قائمة **Run**).

يظهر إطار برنامج السرعة كما هو مبين في الشكل 2-4.





الشكل 2-4

إطار برنامج السرعة.

❑غيّر موضع مؤشّر شريط التمرير بواسطة الفأرة.

كما تلاحظ، لا يظهر أي شيء ضمن مربع النص، والسبب في ذلك طبعاً، هو أننا لم نكتب نص البرنامج اللازم لإظهار القيم الموافقة لتغيير مؤشّر شريط التمرير.

❑انقر الزر **خروج** لإنهاء البرنامج.

الخصائص Min و Max و Value لشريط التمرير

تشرح الفقرات التالية بعض خصائص شريط التمرير:

الخاصيتان Min و Max

يمثّل شريط التمرير مجموعة من القيم. تحدد الخاصية Min القيمة الدنيا، وتحدد الخاصية Max القيمة العليا. مثلاً، تأخذ الخاصية Min في الجدول 2-1 القيمة صفر، وتأخذ الخاصية Max القيمة 100، وهذا يعني أن شريط التمرير يمكن أن يعطي أية قيمة بين الصفر و 100.

الخاصية Value

تمثل الخاصية Value التابعة لشريط التمرير، القيمة الراهنة لهذا الشريط، وبالتالي فقد تكون أية قيمة صحيحة بين الرقم صفر والرقم 100 حسب مثالنا هذا. لم نعط الخاصية Value قيمة معينة أثناء مرحلة التصميم، وبالتالي ستستخدم القيمة الافتراضية (القيمة صفر) لهذه الخاصية، وعند تنفيذ البرنامج، يوضع مؤشّر شريط التمرير عند الموضع المرافق للخاصية Value (أي عند أقصى يمين شريط التمرير، وهو الموضع المرافق للقيمة صفر للخاصية Value).

الآن، وباعتبار أن السرعة الافتراضية يجب أن تكون 50، لهذا يجب إسناد القيمة 50 للخاصية Value التابعة لشريط التمرير:

- أسند القيمة 50 للخاصية Value.

الآن، ستجد عند تنفيذ البرنامج، أن الموضع الافتراضي لمؤشّر شريط التمرير سيكون في وسطه (أي منتصف المسافة بين صفر و 100).

لاحظ أن الجدول 2-1 يطالبك بإسناد القيمة "50 كم / الساعة" للخاصية Text التابعة لمربع النص. فعند تشغيل البرنامج، ستجد أن مربع النص يُظهر القيمة الابتدائية "50 كم / الساعة" والمشابهة للموضع الراهن لمؤشّر شريط التمرير (Value = 50).

تركيز Focus لوحة المفاتيح

تستطيع ضغط المفتاح Tab من على لوحة المفاتيح، أثناء عمل البرنامج، لنقل التركيز من عنصر تحكم إلى آخر. وتستطيع تمييز عنصر التحكم الذي يمتلك التركيز بسهولة، لأن ويندوز يعطي دلالة على ذلك، (توضّع الإضاءة عنده، أو يظهر حول عنوانه مستطيل منقط .. الخ).

فمثلاً، يظهر مؤشّر وامض في مربع النص، إذا كان التركيز موضوعاً عنده، بينما يومض مؤشّر شريط التمرير عندما يكون التركيز موضوعاً عنده. كما يظهر مستطيل منقط حول عنوان الزر **خروج** (مثلاً) إذا كان هذا الزر يمتلك التركيز، وهكذا.

ما المقصود بأن عنصر تحكم ما، يمتلك التركيز ؟!. المقصود من ذلك، هو أنك تستطيع استخدام لوحة المفاتيح للتحكم به عند امتلاكه للتركيز. جرب مثلاً ما يلي لرؤية ظاهرة تركيز لوحة المفاتيح على أرض الواقع:

- ▫نفّذ برنامج السرعة. (لاحظ أن مؤشّر شريط التمرير يتوضع في الوسط وهذا طبعاً بسبب إسناد القيمة 50 للخاصية Value التابعة لشريط التمرير).
- ▫اضغط المفتاح Tab في لوحة المفاتيح، إلى أن يصل التركيز إلى شريط التمرير، (ستجد أن مؤشّر شريط التمرير يُومض).

يمتلك شريط التمرير الآن التركيز، استخدم مفتاحي الأسهم اليميني واليساري على لوحة المفاتيح لتحريك مؤشّر شريط التمرير. باعتبار أن شريط التمرير يمتلك تركيز لوحة المفاتيح فإن الضغط على مفتاحي الأسهم على لوحة المفاتيح يكافئ نقر زري السهمين اليساري واليميني لشريط التمرير. جرب ضغط المفاتيح Home و End و PgUp و PgDn وراقب النتائج.

- ▫اضغط المفتاح Tab حتى يصل التركيز إلى الزر **خروج**، ثم اضغط مفتاح Space أو مفتاح Enter. وهذا يكافئ نقر الزر **خروج** لإنهاء البرنامج.

لاحظ كم من العمليات تستطيع إنجازها بواسطة برنامج السرعة!. تستطيع تبديل موضع مؤشّر شريط التمرير، وتكبير أو تصغير إطار البرنامج (بسحب حواف الإطار)، ونقل إطار البرنامج بسحب شريط عنوانه، وإنجاز الكثير من مهام ويندوز القياسية الأخرى. الجميل في الموضوع، أنه لا يلزم كتابة أي نص برمجي لإنجاز ذلك. بل لعل ذلك من أهم محاسن كتابة برامج تحت بنية ويندوز.

فالمعالم القياسية لويندوز تكون مبرمجة أصلاً في برامجك، ولست بحاجة كمستخدم ويندوز (أو مستخدم لبرامجك)، الإلمام بكل المظاهر القياسية لويندوز حتى تتمكن من العمل بكفاءة وفق هذه البنية.

تحسين برنامج السرعة

سنحسّن برنامج السرعة الآن:

- انقر نقرة مزدوجة على عنصر تحكم شريط التمرير ضمن النموذج لإظهار الإجراء hsbSpeed_Change(). تحقق بأنّ مربع السرد في الزاوية اليسرى العليا من إطار نص البرنامج يحمل العبارة النصية hsbSpeed، وأنّ مربع السرد المجاور له، يحمل العبارة النصية Change، وبالتالي، عند قراءتهما سوياً تنتج العبارة hsbSpeed_Change().
- أدخل النص التالي ضمن الإجراء hsbSpeed_Change():

```
Private Sub hsbSpeed_Change()
   txtSpeed.Text = Str(hsbSpeed.Value) + " كم / ساعة "
End Sub
```

ينفذ الإجراء hsbSpeed_Change() (حسب ما يتبدى من اسمه)، عند تغيير موضع مؤشّر شريط التمرير. وبالتالي تتغير الخاصية Value تلقائياً تبعاً لذلك التغيير. فمثلاً، تُصبح قيمة الخاصية Value مساوية الصفر، عند وضع مؤشّر شريط التمرير عند أقصى يمين الشريط، وذلك بسبب إسناد القيمة صفر إلى الخاصية Min.

ملاحظة

تنتقل بداية شريط التمرير من الجهة اليسرى إلى الجهة اليمنى، عند إسناد القيمة True للخاصية RightToLeft. لمزيد من المعلومات عن الخاصية RightToLeft اقرأ الفصل الثاني والعشرين، (إنشاء تطبيقات عربية السمة مع فيجول بيسك).

يتوجب على مربع النص إظهار قيمة الموضع الجديد لمؤشّر شريط التمرير، عند تغيير موقعه. أي بكلمة أخرى، يلزمنا إسناد قيمة الخاصية Value لشريط التمرير، إلى الخاصية Text لمربع النص، وهذا هو دور العبارة التالية:

```
   txtSpeed.Text = Str(hsbSpeed.Value) + " كم / ساعة "
```

فمثلاً، إذا كانت قيمة الخاصية Value لشريط التمرير تساوي 20، فقيمة الخاصية Text لمربع النص ستساوي 20 كم / ساعة.

تتوقع الخاصية Text أن يُسند لها قيمة نصية (سلسلة كتابية)، أما الخاصية Value فهي عبارة عن قيمة عددية. مما يعني أنّه يجب استخدام التابع الوظيفي ()Str لتحويل القيمة العددية للخاصية Value إلى سلسلة كتابية. يُكتب ضمن قوسي التابع الوظيفي ()Str، القيمة العددية المطلوب تحويلها إلى سلسلة كتابية، فمثلاً، يُستخدم التابع (11)Str لتحويل العدد 11 إلى السلسلة الكتابية "11". كما يُستخدم التابع (12345)Str لتحويل العدد 12345 إلى السلسلة الكتابية "12345".

ملاحظة

عند تحويل القيمة الرقمية إلى قيمة نصية، فإنها تفقد قيمتها الرقمية، وتصبح كأي حرف آخر. طبعاً، يوجد تابع وظيفي معاكس للوظيفة ()Str وهو التابع ()Val، الذي يحول القيمة النصية ("1234") إلى قيمة عددية (1234).

في حالتنا هذه، يُطلب من الإجراء تحويل القيمة العددية hsbSpeed.Value إلى سلسلة كتابية، ولهذا استخدمنا العبارة التالية:

```
Str(hsbSpeed.Value)
```

وبالتالي، فإذا فرضنا مثلاً، أنّ الموضع الحالي لمؤشّر شريط التمرير يساوي 32 (hsbSpeed.Value = 32)، والعبارة التالية:

```
txtSpeed.Text = Str(hsbSpeed.Value) + " كم / ساعة "
```

تُسند للخاصية Text التابعة لمربع النص، القيمة التالية: 32 كم / ساعة

والآن لنشاهد نتائج ما كتبناه على أرض الواقع:

- احفظ المشروع باختيار **Save Project** من القائمة **File**.
- نفّذ برنامج السرعة.
- انقل مؤشّر شريط التمرير، وستجد أنّ محتويات مربع النص تتغير تبعاً لموضع مؤشّر الشريط.
- أنهِ برنامج السرعة بنقر الزر **خروج**.

تغيير محتويات مربع النص مع سحب مؤشّر شريط التمرير

أنهينا برنامج السرعة تقريباً، ولكن بقيت مشكلة واحدة ينبغي حلها. وللتعرف على هذه المشكلة، اتبع الخطوات التالية:

- نفّذ برنامج السرعة.
- جرّب سحب مؤشّر شريط التمرير (دون إفلاته)، وستجد أنّ محتوى مربع النص لا يتغير أثناء عملية السحب! وإنما يتغير فقط بعد تحرير المؤشّر.

كم سيكون جميلاً لو يترافق تغير محتوى مربع النص مع حركة مؤشّر شريط التمرير.

يُنفذ الإجراء hsbSpeed_Scroll() آلياً، عند سحب مؤشّر شريط التمرير، لهذا:

- انقر نقرة مزدوجة على عنصر تحكم شريط التمرير ضمن النموذج، لإظهار الإجراء hsbSpeed_Scroll() في مربع السرد الموجود في الزاوية اليسرى والعليا من إطار نص البرنامج ووجود عبارة النص Scroll في مربع السرد المجاور له.
- أدخل النص التالي في الإجراء hsbSpeed_Scroll():

```
Private Sub hsbSpeed_Scroll()
    hsbSpeed_Change
End Sub
```

- احفظ المشروع باختيار **Save Project** من القائمة **File** لفيجول بيسك.

النص الذي أدخلناه في الإجراء hsbSpeed_Scroll() هو التالي:

```
    hsbSpeed.Change
```

يتسبب نص الإجراء هذا، بتنفيذ الإجراء hsbSpeed_Change()، الذي كتبناه مسبقاً، مما يعني أنّ نص الإجراء hsbSpeed_Change() سينفذ عند سحب مؤشّر شريط التمرير، الذي يعمل على إسناد الموقع الحالي لشريط التمرير إلى الخاصية Text لمربع النص، دعنا نشاهد بأنفسنا أثر ما كتبناه:

- نفّذ برنامج السرعة.
- اسحب مؤشّر شريط التمرير، وتحقق أن محتويات مربع النص، تتغير تبعاً لسحب مؤشّر شريط التمرير.

▫انقر الزر **خروج** لإنهاء برنامج السرعة.

ملاحظة

ينفذ الإجراء ()hsbSpeed_Change من ضمن الإجراء ()hsbSpeed_Scroll كما يلي:
لاحظ، عدم استخدام الأقواس بعد الكلمة hsbSpeed_Change، أي عند استدعاء الإجراء ()hsbSpeed_Change.
سيؤدي استخدام القوسين بعد اسم الإجراء في مثل هذه الحالة إلى ظهور رسالة خطأ. وبالواقع يُظهر فيجول بيسك جزء البرنامج الذي يحمل الخطأ بلون أحمر، وهي دلالة مرئية على وجود خطأ ضمن نص البرنامج.

كلمة أخيرة حول برنامج السرعة

يوضح برنامج السرعة، كيف تتمكن من تشكيل واجهة مستخدم محكمة، لإدخال الأعداد.
فبدلاً من إجبار المستخدم على إدخال الأعداد بين صفر و 100 يدوياً، قدمنا للمستخدم شريط تمرير، يمكّن المستخدم من إعطاء أي قيمة صحيحة ضمن المجال المسموح باستخدام شريط التمرير هذا، والحصول على تغذية عكسية (أو تغذية راجعة Feedback وهي الاستجابة الناتجة عن مربع النص) عن مجال الأرقام المسموح بإدخالها.

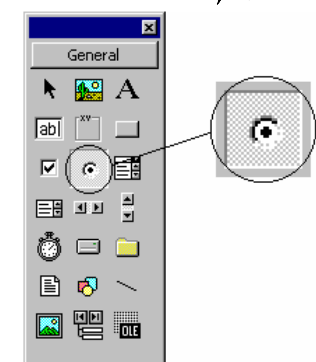
برنامج الخيارات

يوضّح هذا البرنامج كيف يمكنك كتابة برامج تسمح للمستخدم بانتقاء خيار ما Options.

التمثيل المرئي لبرنامج الخيارات

يستخدم برنامج الخيارات عنصر تحكم زر الخيار Options Button. انظر الشكل 2 – 5 لرؤية شكل وموقع عنصر التحكم هذا ضمن مربع الأدوات.

علماً بأن الموقع يتغير حسب الإصدار المستخدم، يؤدي وضع مؤشّر الفأرة فوق أي عنصر تحكم ضمن مربع الأدوات إلى ظهور مستطيل أصغر يحمل بداخله اسم ذلك العنصر (مثلاً Options Button في حالتنا هذه).



الشكل 2-5

رمز زر الخيار Options

ضمن إطار مربع الأدوات.

□أنشئ مشروعاً جديداً باختيار **New Project** من القائمة **File** لفيجول بيسك، ثم اختر الرمز Standard EXE وانقر الزر **فتح** ضمن الإطار New Project.

□تحقق بأن إطار النموذج Form1 هو الإطار الراهن (أي أنه تم اختياره، أو بكلمة أخرى، الإضاءة متوضعة لديه). ثم اختر **Save Form1 As** من القائمة **File** لفيجول بيسك. استخدم الآن مربع الحوار Save File As لحفظ الملف بالاسم Options.Frm في الدليل C:\VB5Prg\Ch02.

□اختر الآن **Save Project As** من القائمة **File** لفيجول بيسك، واستخدم مربع الحوار Save Project لحفظ المشروع بالاسم Options.Vbp في الدليل C:\VB5Prg\Ch02.

□أنشئ النموذج frmOptions طبقاً للجدول 2-2.

يفترض أن يبدو النموذج لدى اكتماله كذاك المبين في الشكل 2-6.



الشكل 2-6
النموذج frmOptions
في طور التصميم.

الجدول 2-2. جدول خصائص النموذج frmOptions.

القيمة	الخاصية	الكائن
frmOptions	**Name**	**Form**
Red	BackColor	
برنامج الخيارات	Caption	
True	RightToLeft	
cmdExit	**Name**	**CommandButton**
&خروج	Caption	
True	RightToLeft	
chkSound	**Name**	**Check Box**
Red	BackColor	
&أصوات	Caption	
(اختر ما شئت)	Font	
White	ForeColor	
True	RightToLeft	
chkMouse	**Name**	**Check Box**
Red	BackColor	
ال&فأرة	Caption	
(اختر ما شئت)	Font	
White	ForeColor	
True	RightToLeft	

الكائن	الخاصية	القيمة
Check Box	**Name**	**chkColors**
	BackColor	Red
	Caption	ال&ألوان
	Font	(اختر ما شئت)
	ForeColor	White
	RightToLeft	True
Option Button	**Name**	**optLevel1**
	BackColor	Red
	Caption	المستوى &1
	Font	(اختر ما شئت)
	ForeColor	White
	RightToLeft	True
Option Button	**Name**	**optLevel2**
	BackColor	Red
	Caption	المستوى &2
	Font	(اختر ما شئت)
	ForeColor	White
	RightToLeft	True
Option Button	**Name**	**optLevel3**
	BackColor	Red
	Caption	المستوى &3
	Font	(اختر ما شئت)
	ForeColor	White
	RightToLeft	True
Label	**Name**	**lblChoice**
	Alignment	2-Center
	BorderStyle	1-Fixed Single
	Font	(اختر ما شئت)
	RightToLeft	True

ملاحظة

ستضطر غالباً إلى زيادة ارتفاع النموذج frmOptions عند بنائه طبقاً للجدول 2–2 وذلك حتى تتسع كل العناصر فيه. ولزيادة الارتفاع اسحب الحافة السفلى للإطار باتجاه الأسفل.

قسم التصاريح العامة General Declarations للنموذج

سنُدخل في هذا القسم جزءاً من برنامج، علماً بأنّ هذا القسم عبارة عن منطقة ضمن إطار نص البرنامج يكتب فيها شتى العبارات العامة.

تعتبر العبارة Option Explicit مثالاً على عبارة عامة. سنتناول المعنى الدقيق لهذه العبارة لاحقاً في هذا الفصل. أما الآن فيكفينا تعلم كيفية تناول قسم التصاريح العامة، وكيفية كتابة نص برمجي داخله.

اتبع الخطوات التالية:

- انقر نقراً مزدوجاً على أي منطقة خالية من النموذج frmOptions لإظهار إطار نص البرنامج (Code Window).

يستجيب فيجول بيسك بإظهار إطار نص البرنامج.

- انقر على رمز السهم النازل لمربع السرد الموجود في الزاوية اليسرى العليا من إطار نص البرنامج، ثم اختر البند (General) من القائمة.
- انقر رمز السهم النازل لمربع السرد المجاور للمربع السرد السابق، والمتوضع عند الزاوية العليا اليمنى من إطار نص البرنامج، ثم اختر البند (Declarations) منه.

يُظهر إطار نص البرنامج الآن قسم التصاريح العامة General Declarations، حسب ما يوضحه الشكل 2–7.

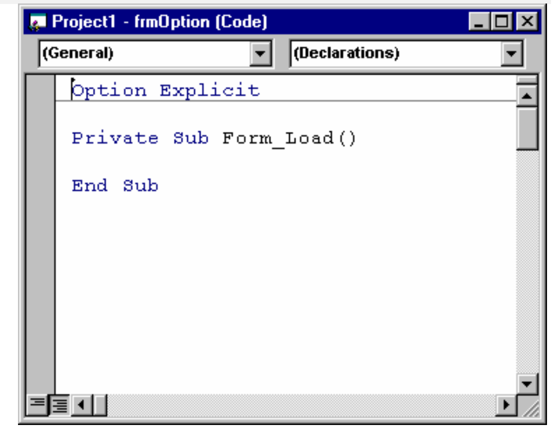
ستُلاحظ النص التالي في الشكل 2–7:

```
Private Sub Form_Load()

End Sub
```

حسب ما هو واضح في الشكل 2-7، يقع قسم التصاريح العامة فوق الإجراء Form_Load()، وقد تجد عبارات مكتوبة مسبقاً في هذا القسم، مثلاً:

```
Option Explicit
```



الشكل 2-7

إظهار قسم التصاريح العامة General Declarations لإطار نص البرنامج Code Window.

□تستطيع الآن نقر قسم التصاريح العامة، وكتابة أي نص برنامج إضافي تريد. فمثلاً إذا لم تشاهد العبارة Option Explicit في هذا القسم، فاكتب العبارة التالية:

```
Option Explicit
```

ربط حادثة Click للزر خروج بنص البرنامج المناسب

□أدخل النص التالي ضمن الإجراء cmdExit_Click() للنموذج frmOptions:

```
Private Sub cmdExit_Click()
    End
End Sub
```

يُنفذ جزء البرنامج الذي أدخلته، آلياً عند نقر الزر **خروج**. علماً بأنّ جزء البرنامج هذا يُنهي عمل برنامج الخيارات.

تنفيذ برنامج الخيارات

رغم أننا لم ننته بعد من تصميم برنامج الخيارات لكن دعنا ننفذه:

□نَفّذ برنامج الخيارات.

□انقر زر الخيار **المستوى1**.

*يستجيب البرنامج، باختيار زر الخيار **المستوى1**، (تظهر دائرة مصمتة ضمن زر*

*الخيار **المستوى1**).*

□انقر زر الخيار **المستوى2.**

*يستجيب البرنامج بإلغاء اختيار زر الخيار **المستوى1** (إزالة الدائرة المصمتة من زر الخيار **المستوى1**)، واختيار زر الخيار **المستوى2** بدلاً عنه (وضع دائرة مصمتة داخله).*

□انقر زر الخيار **المستوى3.**

*يستجيب البرنامج بإلغاء اختيار الزر **المستوى2** ويختار بدلاً منه الزر **المستوى3.***

إذاً يسمح فقط باختيار زر خيار واحد في نفس الوقت. تستخدم أزرار الخيارات ضمن البرامج عندما يرغب المستخدم باختيار خيار واحد فقط من أجل عدة خيارات. (لاحظ أنّ بعض كتب ويندوز، تُطلق على هذا الزر اسم الزر الراديوي Radio Button، تشبيهاً له بأزرار الراديو التي لا يسمح بضغط أكثر من زر واحد في نفس الوقت).

□انقر خانة الاختيار **أصوات.**

*يستجيب البرنامج بوضع علامة اختيار في خانة الاختيار **أصوات.***

□انقر خانات الاختيار الأخرى.

كما تشاهد، يُسمح باختيار أكثر من خانة اختيار واحدة في نفس الوقت. استخدم خانات الاختيار، عندما ترغب في أن يتمكن المستخدم من اختيار عدة وضعيات في نفس الوقت. فمثلاً. قد يختار المستخدم إذا كان البرنامج عبارة عن لعبة!، اللعب مع صوت أو بدونه، وبفأرة أو بدونها، وبألوان أو بدون ألوان.

كما يستطيع المستخدم اللعب وفق مستوى أول أو ثاني أو ثالث، لكن لا حكمة من وراء اللعب بثلاثة مستويات مثلاً دفعة واحدة (عملياً، لا يمكن اللعب بثلاث مستويات في نفس الوقت، حسب المثال).

لإلغاء اختيار خانة الاختيار، انقر مجدداً عليه، فتزول علامة الاختيار من داخله.

□أنهِ البرنامج بنقر الزر **خروج.**

التحقق من أزرار الاختيار وخانات الاختيار التي يتم اختيارها

سنكتب الآن نصاً يتحقق من أزرار الخيار وخانات الاختيار التي يتم اختيارها.

- أدخل النص التالي ضمن الإجراء chkColors_Click() للنموذج frmOptions:

```
Private Sub chkColors_Click()
    UpdateLabel
End Sub
```

- أدخل النص التالي ضمن الإجراء chkMouse_Click():

```
Private Sub chkMouse_Click()
    UpdateLabel
End Sub
```

- أدخل النص التالي ضمن الإجراء chkSound_Click():

```
Private Sub chkSound_Click()
    UpdateLabel
End Sub
```

- أدخل النص التالي ضمن الإجراء optLevel1_Click():

```
Private Sub optLevel1_Click()
    UpdateLabel
End Sub
```

- أدخل النص التالي ضمن الإجراء optLevel2_Click():

```
Private Sub optLevel2_Click()
    UpdateLabel
End Sub
```

- أدخل النص التالي ضمن الإجراء optLevel3_Click():

```
Private Sub OptLevel3_Click()
    UpdateLabel
End Sub
```

ما الذي فعلناه خلال الخطوات السابقة هذه ؟! لقد كتبنا العبارة التالية:

```
    UpdateLabel
```

في جميع الإجراءات السابقة. يقصد بالعبارة UpdateLabel اسم إجراء جديد، وسنباشر بعد قليل بكتابته. يُنفذ هذا الإجراء آلياً فور اختيار أي من أزرار الخيار أو خانات الاختيار في هذا المثال:

اتبع الآن الخطوات التالية لإضافة الإجراء UpdateLabel إلى النموذج:

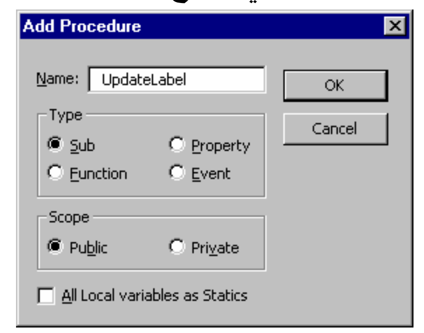
- انقر نقراً مزدوجاً في أي منطقة خالية من النموذج.

يستجيب فيجول بيسك بإظهار إطار نص البرنامج (Code Window).

- اختر Add Procedure من القائمة Tool لفيجول بيسك.

يستجيب فيجول بيسك بإظهار مربع الحوار Add Procedure.

- أدخل UpdateLabel في الحقل Name من مربع الحوار Add Procedure. (لأن UpdateLabel هو اسم الإجراء الجديد الذي سنضيفه). تحقق الآن بأن أزرار الخيارات في مربع الحوار Add Procedure مطابقة للشكل 2-8.



الشكل 2-8

إضافة الإجراء الجديد

.UpdateLabel

- انقر الزر **OK** في مربع الحوار Add Procedure.

يستجيب فيجول بيسك بإظهار إطار نص البرنامج عند الإجراء UpdateLabel جاهزاً للتعديل.

ملاحظة

لقد أضاف فيجول بيسك الإجراء UpdateLabel في المنطقة العامة General Area من النموذج frmOptions. ونستطيع التحقق من ذلك كما يلي:

– ضع قراءة مربع السرد الواقع أعلى يسار إطار نص البرنامج على General.

– ضع قراءة مربع السرد الواقع أعلى يمين إطار نص البرنامج علىDeclarations.

إذا تفقدت محتوى مربع السرد اليميني فستجد بندين فيه: وهما Declarations وUpdateLabel. يوجد لدينا في قسم التصاريح العامة General Declarations العبارة Option Explicit.

الإجراءات التي تضاف إلى النموذج (مثالنا الإجراء UpdateLabel)، تضاف إلى

المنطقة العامة General Area.

من الهام طبعاً معرفة أين تضاف الإجراءات للرجوع إليها لاحقاً بغية قراءتها أو تعديلها.

إذاً، تضاف الإجراءات في المنطقة العامة من النموذج. وللوصول إلى إجراء ما، ضع قراءة مربع السرد الواقع أعلى يسار إطار نص البرنامج على General ثم ضع قراءة مربع السرد اليميني المجاور على اسم الإجراء الذي تريد تعديله أو قراءته.

يكتب فيجول بيسك نيابة عنك، السطرين الأول والأخير من الإجراء، ويبقى عليك كتابة نص الإجراء المناسب:

□أدخل النص التالي ضمن الإجراء UpdateLabel:

```
Public Sub UpdateLabel()
    Dim Info
    Dim LFCR
    LFCR = Chr(13) + Chr(10)
    ' الصوت
    If chkSound.Value = 1 Then
       Info = "الصوت : تشغيل"
    Else
       Info = "الصوت : إيقاف"
    End If
    ' الفأرة
    If chkMouse.Value = 1 Then
       Info = Info + LFCR + "الفأرة : تشغيل"
    Else
       Info = Info + LFCR + "الفأرة : إيقاف"
         f' الألوان
    If chkColors.Value = 1 Then
       Info = Info + LFCR + "الألوان : تشغيل"
    Else
       Info = Info + LFCR + "الألوان : إيقاف"
    End If
    ' المستوى 1
```

```
If optLevel1.Value = True Then
   Info = Info + LFCR + "1 : المستوى"
End If
'
If optLevel2.Value = True Then
   Info = Info + LFCR + "2 : المستوى"
End If
If optLevel3.Value = True Then
   Info = Info + LFCR + "3 : المستوى"
End If

lblChoice.Caption = Info
End Sub
```

❑احفظ المشروع باختيار **Save Project** من القائمة **File**.

تنفيذ برنامج الخيارات

دعنا نُنفّذ البرنامج قبل المضي في دراسة نص الإجراء UpdateLabel:

❑نفّذ برنامج الخيارات.

❑انقر خانات الاختيار وأزرار الخيار المختلفة.

يستجيب برنامج الخيارات، بإظهار حالة خانات الاختيار وأزرار الخيارات ضمن اللافتة lblCHoice (انظر الشكل 2-9).

❑أَنهِ برنامج الخيارات بنقر الزر **خروج**.



الشكل 2-9
إطار برنامج الخيارات
مع حالة التوضعات الحالية
لكل من أزرار الخيارات
وخانات الاختيار.

كيف يعمل برنامج الخيارات

يُنفّذ برنامج الخيارات، الإجراء UpdateLabel عند نقر أحد أزرار الخيارات أو خانات الاختيار.

نص برنامج الإجراء chkColors_Click() للنموذج frmOptions

يُنفّذ هذا الإجراء آلياً، عند نقر خانة الاختيارchkColors:

```
Private Sub chkColors_Click()
    UpdateLabel
End Sub
```

ينفذ بعد ذلك نص الإجراء UpdateLabel والذي سنشرحه بعد قليل.

بطريقة مشابهة، يؤدي نقر أي زر من أزرار الخيارات أو خانات الاختيار الأخرى إلى تنفيذ الإجراء المرافق لذلك العنصر (زر خيار أو خانة اختيار)، وبالتالي تنفيذ الإجراء UpdateLabel.

نص برنامج الإجراء UpdateLabel

إذاً، ينفذ هذا الإجراء حسب ما ذكرنا، عند اختيار أي من أزرار الخيارات أو خانات الاختيار في هذا المثال.

هذا الإجراء ليس إجراء حادثة خاصة بأحد كائنات فيجول بيسك، أي لا ينفذ تلقائياً عند وقوع حادثة ما، وإنما هو إجراء أنشأناه بأنفسنا باستخدام مربع الحوار Add Procedure كما مرّ معنا سابقاً.

من الهام التمييز بين إجراءات مثل cmdExit_Click() (إجراءات مرتبطة بحادثة ما)، وبين إجراءات مثل UpdateLabel، فالإجراء cmdExit_Click() ينفذ آلياً عند وقوع حادثته، ولا حاجة لكتابة أي نص برمجي للتسبب بعملية تنفيذه. ما هو السبب؟!. السبب أنّ فيجول بيسك يعمل!، ففي اللحظة التي يتم فيها نقر الزر **خروج**. تقع حادثة النقر Click، وبالتالي ينفذ الإجراء cmdExit_Click().

لا ينفذ الإجراء UpdateLabel آلياً، بل يجب على برنامجك أن يستدعيه للتنفيذ، ويتم ذلك بذكر اسمه، وهو السبب الذي دفعنا إلى كتابة العبارة pdateLabel في ستة مواقع، فالعبارة:

```
UpdateLabel
```

تتسبب بتنفيذ الإجراء UpdateLabel. مما يعني أنّ هذا الإجراء سينفذ عند نقر أي من أزرار الخيار أو خانات الاختيار في هذا المثال.

التصريح عن المتحول Info

تصرح أول عبارة كتبناها في الإجراء UpdateLabel عن المتحول Info بالشكل التالي:

```
Dim Info
```

تعتبر Dim تعليمة فيجول بيسك، وتدل أن الكلمة التي تليها (Info في مثالنا هذا)، هي اسم لمتحول سوف نستخدمه لاحقاً في الإجراء. يُستخدم المتحول Info كمتحول نصي، يقوم بتخزين سلسلة من الأحرف الكتابية أثناء تنفيذ الإجراء UpdateLabel.

تستطيع التصريح عن هذا المتحول بالطريقة التالية أيضاً:

```
Dim Info As String
```

لحسن الحظ، يعتبر فيجول بيسك متساهلاً من هذه الناحية، ولا يُجبرك على التصريح عن نوع المتحول، بل يفترض نوعه حسب طريقة استخدامه.

يعرف من لديه شيئاً من الخبرة في لغات البرمجة الأخرى، أنّ بعض لغات البرمجة لا تتطلب من المستخدم التصريح عن المتحولات، لكن تبقى عادةُ التصريح عن المتحولات عادة حسنة، ولمعرفة السبب افترض أنّ الإجراء يحوي على الحسابات التالية:

```
Time = 10
Velocity = 50
Distance = Velocity * Time
lblDistance.Caption = " = المسافة" + Str(Distance)
```

تُسند العبارات الأربعة السابقة، القيمة 10 إلى المتحول Time، والقيمة 50 إلى المتحول Velocity، ثم تحسب حاصل ضرب هذين المتحولين وتظهر المسافة Distance، بإسناد قيمة المسافة Distance إلى الخاصية Caption للافتة lblDistance.

لنفترض الآن، أنك كتبت أحد المتحولات بشكل خاطئ، (نسيت مثلاً كتابة الحرف a بعد الحرف t في كلمة Distance) كالتالي:

```
lblDistance.Caption = " = المسافة" + Str(Distnce)
```

يَعتبر فيجول بيسك أنّ **Distnce** (المتحول المكتوب بشكل خاطئ)، هو متحول جديد مختلف عن المتحول الأساسي Distance، ويُسند له آلياً القيمة صفر، وبالتالي تُظهر اللافتة lblDistance الجملة التالية:

```
المسافة = 0
```

وهذا بالطبع خطأ جسيم، قد تهدر كثيراً من الوقت لاكتشافه. تستطيع تلافي حدوث أمثال هذه الأخطاء، بأن تدعو فيجول بيسك إلى التذمر عند استخدام متحول في فيجول بيسك بدون التصريح عنه مسبقاً. وبالتالي، في مثالنا هذا يجب أن تؤول العبارات السابقة إلى ما يلي:

```
Dim Time
Dim Velocity
Dim Distance

Time = 10
Velocity = 50
Distance = Velocity * Time
lblDistance.Caption = " = المسافة" + Str(Distnce)
```

إذا تم إعداد فيجول بيسك، بحيث يتذمر لدى مصادفته متحول غير مصرح عنه ضمن نص البرنامج، فإنه سوف يعطيك رسالة خطأ أثناء تنفيذ البرنامج ويخبرك بأن المتحول غير معروف لديه. فمثلاً يضيء فيجول بيسك المتحول Distnce في حالتنا هذه، ويخبرك بأن فيه شيئاً ما خاطئ.

أما كيف يتم إعداد فيجول بيسك للتذمر عند مصادفة متحولات غير مصرح عنها في نص برنامج ؟! فيتم ذلك بوضع العبارة التالية:

```
Option Explicit
```

ضمن قسم التصاريح العامة General Declarations، وهو السبب الذي دفعنا إلى وضع العبارة Option Explicit في قسم التصاريح العامة عبر الأمثلة السابقة.

ملاحظة

ضع العبارة Option Explicit دائماً ضمن قسم التصاريح العامة للنموذج. فبهذه الطريقة تخبر فيجول بيسك بعدم قبول المتحولات غير المصرح عنها وتوفر على نفسك ساعات طويلة من تنقيح الأخطاء. لنتكلم بعمومية أكثر، يؤدي تجاهل العبارة Option Explicit إلى إنفاق ساعات طويلة في تنقيح أخطاء قد تكون بسيطة ناتجة من أخطاء في التهجي، لهذا كن حكيماً ودع فيجول بيسك يجد عنك المتحولات التي كتبتها بشكل خاطئ.

التصريح عن المتحول LFCR

صرحنا أيضاً عن المتحول LFCR ضمن الإجراء UpdateLabel:

```
Dim LFCR
```

ثم أسندنا إلى المتحول LFCR ما يلي:

```
LFCR = Chr(13) + Chr(10)
```

يمثل الرمز Chr(13) رمز المفتاح Enter على لوحة المفاتيح، كما أنّ الرمز Chr(10) هو رمز التغذية السطرية (أي ينقل مؤشر الكتابة إلى سطر جديد). وكما سنرى، تُظهر اللافتة lblChoice سلسلة طويلة تنتشر على عدة أسطر، وذلك بالاستعانة بالمتحول LFCR.

التحقق من قيمة الخاصية Value

يأتي بعد التصريح عن المتحولات في الإجراء UpdateLabel كتلة الشرط If.Else.End If:

```
'الصوت
If chkSound.Value = 1 Then
   Info = "الصوت : تشغيل"
Else
   Info = "الصوت : إيقاف"
End If
```

تستطيع في فيجول بيسك، إضافة تعليقات ضمن نص البرنامج، باستخدام رمز الفاصلة العلوية (') أو الكلمة Rem. فمثلاً السطر التالي:

```
'الصوت
```

يطابق السطر:

```
Rem الصوت
```

يَستخدم هذا الكتاب الفاصلة العلوية للدلالة على أسطر التعليقات.

كما يمكن إضافة التعليقات في أسطر البرنامج، كما يلي:

```
MyVariable = 1  ' تهيئة المتحول
```

تعتبر عادة وضع التعليقات ضمن نص البرنامج عادة حسنة، لأنها تسهل قراءة وتنقيح البرامج. تستطيع كتابة أي شيء تريده بعد رمز الفاصلة العلوية (')، يتجاهل فيجول بيسك كل الرموز التي تلي هذا الرمز أي (').

يتحقق الإجراء UpdateLabel من أن قيمة الخاصية Value لخانة الاختيار chkSound تساوي 1. فإذا كانت الخاصية Value تساوي واحد، فهذا يعني أنّ العبارات بين السطر If والسطر Else سوف تُنفذ، وفي هذه الحالة لدينا عبارة واحدة فقط بين If و Else وهي العبارة:

```
Info = "الصوت : تشغيل"
```

حيث تنفذ هذه العبارة عندما Value تساوي الواحد. تسند هذه العبارة السلسلة "الصوت : تشغيل" إلى المتحول Info. لاحظ أنه يجب ذكر كلمة Then ضمن العبارة If.

إذا كانت قيمة الخاصية Value لخانة الاختيار **الصوت** تساوي الواحد، فهذا يعني أنه توجد علامة اختيار في خانة الاختيار chkSound؛ وبالتالي سوف تساوي قيمة المتحول Info إلى:

```
"الصوت : تشغيل"
```

تنفذ العبارات الموجودة بين Else و End If، إذا كانت قيمة الخاصية Value لخانة الاختيار chkSound لا تساوي الواحد. فعند عدم وجود علامة اختيار في خانة الاختيار تكون قيمة الخاصية Value لخانة الاختيار مساوية الصفر، وبالتالي تنفذ العبارة الواقعة بين Else و End If، وتسند هذه العبارة إلى المتحول Info الجملة التالية:

```
"الصوت : إيقاف"
```

إذاً لنلخص ما سبق، يُسنَد إلى المتحول Info إما الجملة:

```
"الصوت : تشغيل"
```

أو:

```
"الصوت : إيقاف"
```

بطريقة مشابهة، تتحقق العبارة If.Else.End If بأن قيمة الخاصية Value لخانة الاختيار chkMouse تساوي الواحد:

```
' الفأرة
If chkMouse.Value = 1 Then
   Info = Info + LFCR + "الفأرة : تشغيل"
Else
   Info = Info + LFCR + "الفأرة : إيقاف"
End If
```

فمثلاً، إذا كانت خانة الاختيار chkSound تحوي علامة اختيار داخلها، وكانت خانة الاختيار chkMouse لا تحوي علامة اختيار، فسوف تتسبب عبارتا If.Else.End If الأوليتين من الإجراء UpdateLabel، بإسناد السلسلة التالية إلى المتحول Info:

```
"الصوت : تشغيل" + LFCR + "الفأرة : إيقاف"
```

وسوف تظهر هذه السلسلة لاحقاً على سطرين:

```
الصوت : تشغيل
الفأرة : إيقاف
```

وذلك بسبب إضافة LFCR بين السلسلتين.

تتحقق عبارة If.Else.End If التالية ضمن الإجراء UpdateLabel بأن قيمة الخاصية Value لخانة الاختيار chkColors تساوي الواحد، وتعدّل المتحول Info تبعاً لذلك:

```
' الألوان
If chkColors.Value = 1 Then
```

```
   Info = Info + LFCR + "الألوان : تشغيل"
Else
   Info = Info + LFCR + "الألوان : إيقاف"
End If
```

تتحقق عبارة If.End.If التالية في الإجراء UpdateLabel بأن الخاصية Value لزر الخيار optLevel1 تساوي قيمة الثابت True:

```
If optLevel1.Value = True Then
   Info = Info + LFCR + "المستوى : 1"
End If
```

تحدد الخاصية Value أيضاً حالة عنصر التحكم هذا، فإذا كانت قيمة الخاصية Value تساوي True، فهذا معناه أنه تم اختيار زر الخيار، ويتم إعداد المتحول Info تبعاً لذلك. أما إذا كانت الخاصية Value لزر الخيار optLevel1 لا تساوي True فهذا يعني أنه لم يتم انتقاء زر الخيار هذا.

ملاحظة
الخاصية Value لخانة الاختيار قد تساوي 0 أو 1 أو 2. فإذا كانت تساوي الواحد، فهذا معناه وجود علامة اختيار بداخله. أما إذا كانت تساوي صفر، فهذا يعني عدم وجود علامة اختيار، بينما إذا كانت Value لمربع اختيار تساوي 2، فهذا يعني حالة بين الاثنين، ويظهر بشكل رمادي أو باهت.

ملاحظة
الخاصية Value لزر خيار ما، قد تساوي True أو False. فإذا كانت تساوي True، فهذا معناه وجود نقطة داخل زر الخيار. أما إذا كانت تساوي False فهذا يعني عدم وجود نقطة داخله. وبالتالي عدم اختياره.

تعدّل عبارتا If.End.If التاليتين المتحول Info تبعاً لقيمة الخاصية Value لكل من زري الخيار optLevel2 و optLevel3:

```
If optLevel2.Value = True Then
```

```
   Info = Info + LFCR + "2 : المستوى"
End If
If optLevel3.Value = True Then
   Info = Info + LFCR + "3 : المستوى"
End If
```

تسند آخر عبارة في الإجراء UpdateLabel محتوى المتحول Info إلى الخاصية Caption للافتة lblChoice:

```
lblChoice.Caption = Info
```

تظهر هذه العبارة محتوى المتحول Info داخل اللافتة lblChoice، كما يظهره الشكل2-9.

ماذا لدينا أيضاً ؟!

لا بد أنك أدركت حتى الآن، أنّ البرمجة بلغة فيجول بيسك ترتكز على فهم معنى كل عنصر من عناصر التحكم داخل مربع الأدوات، ومعنى ووظيفة ذلك العنصر. فنفس الخاصية Property، تحمل معاني مختلفة لعناصر التحكم المختلفة.

فمثلاً الخاصية Caption (العنوان) للنموذج، تحوي النص الذي يظهر في شريط عنوان النموذج، أما الخاصية Caption لعنصر التحكم Label (لافتة)، فتحوي النص الذي سيظهر في اللافتة. كذلك الخاصية Value لخانة اختيار، تُشير إلى وجود أو عدم وجود علامة اختيار فيه. والخاصية Value لزر الخيار، تشير إلى وجود أو عدم وجود دائرة مصمتة فيه. وأما الخاصية Value لشريط تمرير، فتشير إلى الموضع الحالي لمؤشّر شريط التمرير.

يحوي مربع الأدوات على رموز عناصر التحكم، ويمتلك كل عنصر تحكم، مجموعته الخاصة من الخصائص Properties. تعتبر بعض عناصر التحكم Controls، عناصر تحكم ويندوز قياسية، مثال ذلك، شريطي التمرير الأفقي والعمودي، ومربعات النص، واللافتات، وخانات الاختيار، وأزرار الخيار، وأزرار الأوامر.

يمكن إضافة رموز أخرى إلى مربع الأدوات في فيجول بيسك، ثم وضع عناصر التحكم التي تمثلها هذه الرموز في النموذج. تدعى الرموز الإضافية لعناصر التحكم هذه بالاسم ActiveX Controls، وتعرف أيضاً باسم عناصر التحكم OCX. سنتعرف بشكل أوسع على هذه العناصر عبر فصول الكتاب.

لا تُعتبر لغة فيجول بيسك لغة برمجة صعبة التعلم، ولكن هنالك الكثير مما يجب تعلمه. ومفتاح التعلم الناجح هو التمرن والتجريب. والمفترض بعد كتابة برامج هذا الكتاب أن تصبح قادراً على تدريس اللغة. فحاول أن تكتب البرامج، وأن تفهم نصوصها، وكذلك جرّب تغيير خصائص عنصر التحكم أثناء طور التصميم، (أي أثناء مرحلة بناء النموذج – مرحلة التمثيل المرئي للبرنامج)، ومرحلة التنفيذ.

يعني تغيير الخصائص أثناء مرحلة التنفيذ، تغيير قيمة الخاصية داخل نص البرنامج. فمثلاً، تسند العبارة الأخيرة في الإجراء UpdateLabel المتحول Info أثناء مرحلة التنفيذ إلى الخاصية Caption للافتة lblCHoice:

```
lblChoice.Caption = Info
```

بينما بالمقابل، أسند العنوان **برنامج الخيارات** إلى الخاصية Caption للنموذج خلال مرحلة التصميم Design Time.

تتقبل بعض الخصائص تغيير قيمتها، خلال أي من الطورين، طور التصميم، أو طور التنفيذ. بينما تقبل بعض الخصائص تغيير قيمتها فقط أثناء مرحلة التنفيذ، (أي من خلال نص البرنامج فقط). فمثلاً تقبل الخاصية Caption لعنصر التحكم Label تغيير قيمتها خلال كلا الطورين: التصميم والتنفيذ.

ستتعرف في هذا الكتاب على خصائص لا يمكن إسنادها أو تغييرها، إلا من خلال طور زمن التنفيذ.

اصطلاحات التسمية المستخدمة في هذا الكتاب

تُسمى عناصر التحكم عبر هذا الكتاب، وفق الجدول 2-3. فتبدأ أسماء أزرار الأوامر مثلاً، بالرموز cmd (كما في cmdMyButton)، وتبدأ أسماء مربعات النص بالرموز txt (كما في txtMyTextBook).

الجدول 2-3. اصطلاحات التسمية لعناصر تحكم فيجول بيسك Controls.

نوع عنصر التحكم	البادئة	مثال
Check box	chk	chkReadOnly
Combo box	cbo	cboEnglish
Command button	cmd	cmdExit
Common dialog	dlg	dlgFileOpen
Communications	com	comFax
Data control	dat	datBiblio
Directory list box	dir	dirSource
Drive list box	drv	drvTarget
File list box	fil	filSource
Form	frm	frmEntry
Frame	fra	fraLanguage
Grid	grd	grdPrices
Horizontal scroll bar	hsb	hsbVolume
Image	img	imgIcon
Label	lbl	lblHelpMessage
Line	lin	linVertical
List box	lst	lstPolicyCodes
MCI	mci	mciVideo
MDI child form	mdi	mdiNote
Menu	mnu	mnuFileOpen
Picture	pic	picVGA
Shape	shp	shpCircle
Text box	txt	txtLastName
Timer	tmr	tmrAlarm
UpDown	upd	updDirection
Vertical scroll bar	vsb	vsbRate

لا تعتبر تسمية الخاصية Name لعناصر التحكم وفق الجدول 2–3 من متطلبات فيجول بيسك. فمثلاً، كنا قد أطلقنا التسمية lblChoice على عنصر التحكم Label الذي يظهر معلومات حول خيار المستخدم. نستطيع إسناد القيمة Choice للخاصية Name لعنصر التحكم بدلاً من lblChoice، وبالتالي ستصبح آخر عبارة في الإجراء UpdateLabel كما يلي:

```
Choice.Caption = Info
```

بدلاً من:

```
lblChoice.Caption = Info
```

لاحظ أنّ تسمية عناصر التحكم وفق الجدول 2–3 تسهل قراءة البرنامج. فإذا نظرت مثلاً إلى العبارة:

```
Choice.Caption = Info
```

لن تكون أنت أو غيرك قادراً على التنبؤ بأن Choice عبارة عن عنصر تحكم Label. بل قد يعتقد من يقرأ هذه العبارة أنها تعديل لعنوان النموذج المسمى Choice. بينما تأمل العبارة:

```
lblChoice.Caption = Info
```

سيدرك من يقرأ هذه العبارة أنّ lblChoice هو اسم لافتة، وهذا بسبب وضع الأحرف lbl. وهكذا يصبح معنى العبارة "أسند قيمة المتحول Info إلى الخاصية Caption للافتة".

العبارات التي قد لا تتسع على سطر واحد في هذا الكتاب

العبارة يمكن إدخالها بحيث تمتد على أكثر من سطر واحد، فمثلاً العبارة:

```
MyVariable = 1 + 2 + 3
```

يمكن كتابتها بهذه الطريقة:

```
MyVariable = 1 + _
             2 + 3
```

أو بهذه الطريقة:

```
MyVariable = 1 _
             + 2 _
             + 3
```

إذاً، نستطيع في فيجول بيسك استئناف كتابة العبارة على السطر التالي وذلك بترك فراغ في نهاية السطر، يليه الرمز (_) Underscore.

ملاحظة

لا تستطيع كتابة العبارة الواحدة على أسطر وأنت في منتصف سلسلة، فمثلاً العبارة التالية:

```
lblMyLabel.Caption = "This is my string "
```

لا يمكن كتابتها كالتالي:

```
lblMyLabel.Caption = "This is _
                     my string "
```

إذا كان لا بُد، فاكتبها كالتالي:

```
lblMyLabel.Caption = "This is _
                  &  "my string "
```

الخلاصة

قفزنا مع هذا الفصل إلى الماء!. أجل لقد بدأنا فعلياً ببناء برامج ويندوز حقيقية تتضمن شريط تمرير ومربع نص ولافته وزر أمر وخانات أختيار وأزرار خيارات. فتعلمنا كيف نضع عناصر التحكم هذه فى برامج لغة فيجول بيسك، وكيفية تحديد قيمة الخاصية Value لهذه العناصر وكيف نضيف نص برنامج إلى الإجراءات المرافقة لعناصر االتحكم هذه. كما تعلمنا من هذا الفصل أيضاً ككيف نضيف إجراء إلى النموذج (عندما أضفنا الإجراء Update Label إلى النموذج FrmOpptioon).

الفصل الثالث

كتل البناء البرمجية

يركز هذا الفصل على كتل البناء البرمجية، فلغة فيجول بيسك كحال اللغات الأخرى، تستخدم كتل بناء برمجية: مثل الإجراءات Procedures، والتوابع الوظيفية Functions، وعبارات الشرط If End If، وعبارات الحلقات، والمتحولات، وغير ذلك من مفاهيم البرمجة الهامة الأخرى.

برنامج ضرب عددين

يوضح هذا البرنامج كيف تستخدم الإجراءات والتوابع الوظيفية ضمن البرامج المطوّرة في لغة فيجول بيسك.

التمثيل المرئي لبرنامج ضرب عددين

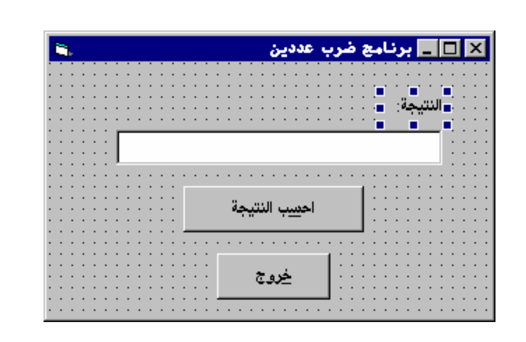
سننجز الآن التمثيل المرئي لبرنامج ضرب عددين:

- أنشئ الدليل C:\VB5Prg\Ch03، سنستخدم هذا الدليل لحفظ العمل المنجز.
- افتح مشروعاً جديداً من نوع Standard EXE، واحفظ نموذج المشروع بالاسم Multiply.frm في الدليل C:\VB5Prg\Ch03، واحفظ ملف المشروع بالاسم Multiply.Vbp في ذات الدليل.
- أنشئ النموذج frmMultiply طبقاً للجدول 3-1.

يفترض أن يظهر النموذج لدى اكتماله كما في الشكل 3-1.

الجدول 3-1. جدول خصائص النموذج frmMultiply.

الكائن	الخاصية	القيمة
Form	**Name**	**frmMultiply**
	Caption	برنامج ضرب عددين
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdCalculate**
	Caption	اح&سب النتيجة
	RightToLeft	True
Text Box	**Name**	**txtResult**
	Text	(اجعله فارغاً)
	RightToLeft	True
Label	**Name**	**lblResult**
	Caption	النتيجة:
	RightToLeft	True



الشكل 3-1

النموذج frmMultiply

بعد انتهاء تصميمه.

إدخال نص برنامج ضرب عددين

سندخل الآن، نص برنامج ضرب عددين:

▫انقر نقراً مزدوجاً على النموذج frmMultiply، لإظهار إطار نص البرنامج.

توجد لائحتين عند قمة إطار نص البرنامج. دع اللائحة اليسارية تشير إلى General، واليمينية إلى Declarations، (وبالتالي فإن مجموع قراءة اللائحتين، سيشير إلى General Declarations، أي قسم التصاريح العامة).

▫اكتب العبارة التالية ضمن قسم التصاريح العامة General Declarations للنموذج frmMultiply:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

يجب من الآن فصاعداً، التصريح عن المتحول قبل استخدامه في البرنامج.

▫أدخل النص التالي ضمن الإجراء cmdCalculate_Click() للنموذج frmMultiply:

```
Private Sub cmdCalculate_Click()
    Multiply 2, 3
End Sub
```

يُنفذ النص الذي أدخلته ضمن الإجراء cmdCalculate_Click() آلياً، عند نقر الزر **احسب النتيجة**. يُنفذ نص البرنامج هذا، الإجراء المدعو Multiply، سنتعلم بعد قليل كيف نضيف الإجراء Multiply إلى برنامجنا.

▫أدخل النص التالي ضمن الإجراء cmdExit_Click() للنموذج frmMultiply:

```
Private Sub cmdExit_Click()
    End
```

```
End Sub
```

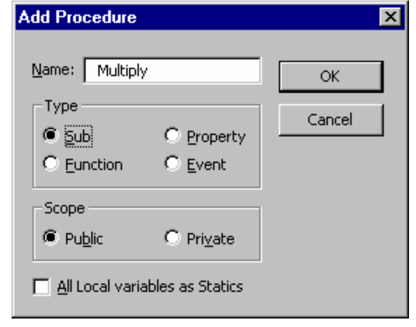
ستضيف إجراء جديداً الآن، إلى النموذج frmMultiply، وسنطلق على هذا الإجراء التسمية Multiply. إليك طريقة إنجاز ذلك:

- انقر نقراً مزدوجاً على النموذج frmMultiply لإظهار إطار نص البرنامج.
- انقر **Add Procedure** من القائمة **Tools** للغة فيجول بيسك.

يستجيب فيجول بيسك بإظهار مربع الحوار Add Procedure.

- أدخل Multiply في مربع النص Name ضمن مربع الحوار Add Procedure.
- ضع علامة بجانب Sub ضمن الحقل Type. (لأنا نوع الإجراء هو روتين فرعي).
- ضع علامة بجانب Public ضمن الحقل Scope. (لأن نريد أن يكون مجال الإجراء، مرئياً لجميع وحدات المشروع).

يبين الشكل 3-2 كيف يبدو مربع الحوار Add Procedure الآن.



الشكل 3-2
إضافة الإجراء
المدعو Multiply.

- انقر الزر **OK** لإغلاق مربع الحوار Add Procedure.

يستجيب فيجول بيسك بإضافة الإجراء Multiply() إلى المنطقة العامة من النموذج frmMultiply، ويظهر نص الإجراء Multiply بالشكل التالي:

```
Public Sub Multiply()
End Sub
```

سنحتاج إلى تعديل السطر الأول من الإجراء Multiply() ليصبح الإجراء بعدها كالتالي:

```
Public Sub Multiply(X As Integer, Y As Integer)
End Sub
```

لأننا سنمرر للإجراء Multiply، وسيطين يمثلان العددين المطلوب معرفة ناتج ضربهما.

□أدخل الآن النص التالي ضمن الإجراء Multiply:

```
Public Sub Multiply(X As Integer, Y As Integer)
    Dim Z
    Z = X * Y
    txtResult.Text = Str(Z)
End Sub
```

□اختر البند **Save Project** من القائمة **File** لفيجول بيسك لتحفظ عملك.

تنفيذ برنامج ضرب عددين

□نفّذ برنامج ضرب عددين، وانقر الزر **احسب النتيجة**.

*يستجيب البرنامج بإظهار الرقم 6 ضمن مربع النص. (يحسب الإجراء ناتج ضرب 2 مع 3، ويظهر الناتج ضمن مربع النص، عند نقر الزر **احسب النتيجة**).*

كيف يعمل برنامج ضرب عددين

يُنفذ برنامج ضرب عددين، الإجراء ()Multiply لحساب حاصل ضرب عددين.

نص الإجراء ()cmdCalculate_Click

ينفذ هذا الإجراء عند نقر الزر **احسب النتيجة**. تَستدعي العبارة الوحيدة الموجودة ضمنه الإجراء ()Multiply مع وسيطين هما العددين 2 و 3:

```
Private Sub cmdCalculate_Click()
    Multiply 2, 3
End Sub
```

يحتاج الإجراء لمعرفة العددين اللذين سيحسب حاصل ضربهما، ولهذا قدمنا له العددين 2 و3 كوسيطين له.

نص الإجراء ()Multiply

يمتلك الإجراء ()Multiply وسيطين:

يدعى الوسيط الأول X، ويصرح عنه بأنه من نوع صحيح Integer ، والوسيط الثانيY من

النوع صحيح Integer أيضاً:

```
Public Sub Multiply(X As Integer, Y As Integer)
    Dim Z
    Z = X * Y
    txtResult.Text = Str(Z)
End Sub
```

يُصرح الإجراء ()Multiply عن متحول يدعى Z ويُسند إليه ناتج حاصل ضرب X مع Y.

ثم يستخدم الإجراء، المتحول Z بعدها، لإسناد قيمة Z إلى الخاصية Text لمربع النص txtResult. وهذا طبعاً بعد تحويل قيمة Z إلى ثابت كتابي بواسطة التابع الوظيفي ()Str.

يمكن التصريح عن إجراء ما، في فيجول بيسك، دون استخدام أي وسائط، فيكون القوسان بعد اسم الإجراء فارغتان، وبالتالي لا يلزم تحديد وسائط عند استدعاء الإجراء، فمثلاً لدينا الإجراء:

```
Public Sub Updatelabel()
```

يُستدعى هذا الإجراء بالطريقة التالية:

```
UpdateLabel
```

أي يكفي ذكر اسمه.

بينما استخدمنا في مثالنا الحالي، هذه العبارة:

```
Multiply  2,3
```

لاستدعاء (تنفيذ) الإجراءات في لغة فيجول بيسك، فمثلاً نستطيع إعادة كتابة الإجراءات ()cmdCalculate_Click بالشكل التالي:

```
Private Sub cmdCalculate_Click()
    Call Multiply (2,3)
End Sub
```

ملاحظة

يجب استخدام الأقواس للإحاطة بوسائط الإجراء، عند استدعائه بواسطة العبارة Call. لا تهم الطريقة التي تُستدعى بها الإجراء، ولهذا يمكنك اختيار الطريقة التي تناسبك. لدى إضافة الإجراء Multiply() اخترنا النطاق Scope بأنه من نوع Public (انظر الشكل 3−2). ولهذا صرح فيجول بيسك عن الإجراء Multiply بأنه من النوع العام Public:

```
Public Sub Multiply()
```

لو اخترنا النطاق Private بدلاً من Public، لكان شكل التصريح كما يلي:

```
Private Sub Multiply()
```

الفرق بين التصريحين، أن الإجراء في حالة Public يكون عاماً، ويمكن استخدامه من قبل أي إجراء آخر في أي موقع من ملفات المشروع، فحسب ما سيتوضح لك عبر الكتاب يمكن للبرنامج الواحد أن يتألف من عدة ملفات (ربما عدة نماذج) وبالتالي، فعند التصريح عن إجراء بأنه Public، فإنّ بمقدور كل ملفات البرنامج استخدامه.

أما عند استخدام Private، فتنعكس الآية، إذ لا يمكن استخدام الإجراء إلا من قبل الإجراءات الموجودة ضمن نفس الملف. وفي حالتنا هذه يمكن لإجراءات النموذج frmMultiply فقط استخدام الإجراء Multiply().

استخدام تابع وظيفي داخل برنامج ضرب عددين

يُطلق على الإجراء Multiply() عبارة إجراء Procedure، لأنه لا يعيد أي قيمة. وهو يشبه التابع الوظيفي أي Function، باستثناء أنه يعيد قيمة بعد استدعائه.

سنكتب الآن نص برنامج يوضح كيفية استخدام التابع الوظيفي. أولاً، احذف الإجراء Multiply() من برنامج ضرب عددين باتباع ما يلي:

- أظهر إطار نص البرنامج بالنقر المزدوج على منطقة خالية في النموذج frmMultiply.
- حدد موقع الإجراء Multiply() في إطار نص البرنامج، ستعثر على هذا الإجراء

ضمن المنطقة General Declarations، والتي تستطيع الانتقال إليها بوضع اللائحة اليسارية الموجودة عند قمة إطار نص البرنامج على General ووضع اللائحة المجاورة لها على Multiply.

- ضع إضاءة فوق كامل الإجراء. بما في ذلك عنوانه وآخر سطر فيه، (ضع المؤشر عند أول حرف في الإجراء، ثم اضغط Ctrl+A).
- اضغط المفتاح Delete على لوحة المفاتيح.

هذا كل شيء بالنسبة لحذف الإجراء Multiply.

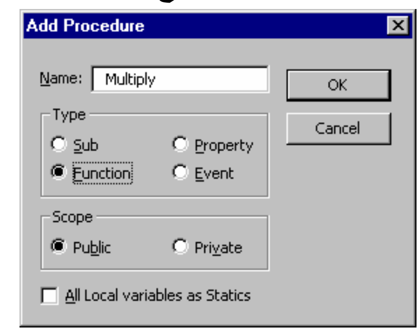
اتبع الآن الخطوات التالية لإضافة تابع وظيفي جديد إلى البرنامج Multiply():

- أظهر إطار نص برنامج النموذج frmMultiply.
- اختر **Add Procedure** من القائمة **Tools**.

يستجيب فيجول بيسك بإظهار مربع الحوار Add Procedure.

- أدخل الاسم Multiply ضمن مربع النص Name.
- اختر Function ضمن الحقل Type.
- اختر Public ضمن الحقل Scope.

يفترض أن يبدو مربع الحوار Add Procedure الآن كما في الشكل 3-3.



الشكل 3-3

مربع الحوار Add Procedure

وإضافة التابع الوظيفي Multiply().

- انقر الزر **Ok** لإغلاق مربع الحوار Add Procedure.

يستجيب فيجول بيسك بإضافة التابع الوظيفي Multiply() إلى قسم التصاريح العامة General Declarations للنموذج frmMultiply ويظهر إطار نص البرنامج مع التابع الوظيفي Multiply() جاهزاً للتعديل:

```
Public Function Multiply()
End Function
```

□غير السطر الأول لهذا التابع بحيث يغدو كالتالي:

```
Public Function Multiply(X As Integer, Y As Integer)
End Function
```

والآن يمتلك التابع الوظيفي Multiply() وسيطين (كلاهما من نوع صحيح Integer):

□أضف النص التالي إلى التابع الوظيفي Multiply():

```
Public Function Multiply(X As Integer, Y As Integer)
    Dim Z
    Z = X * Y
    Multiply = Z
End Function
```

□غيّر نص الإجراء cmdCalculate_Click() ليصبح كما يلي:

```
Private Sub cmdCalculate_Click()
    txtResult.Text = Str(Multiply(2,3))
End Sub
```

□احفظ المشروع باختيار **Save Project** من القائمة **File**.

□نفّذ برنامج ضرب عددين.

□انقر الزر **احسب النتيجة**.

يتصرف البرنامج كما ترى بذات الطريقة السابقة (في حالة الإجراء Multiply()).

□أنهِ البرنامج بنقر الزر **خروج**.

نص التابع الوظيفي Multiply()

يصرح ضمن هذا التابع عن متحول يدعىZ ويسند إليه ناتج حاصل ضرب الوسيطينX وY:

```
Public Function Multiply(X As Integer, Y As Integer)
    Dim Z
    Z = X * Y
    Multiply = Z
End Function
```

تُسند آخر عبارة في التابع الوظيفي Multiply() قيمة المتحول Z إلى المتحول Multiply:

```
    Multiply = Z
```

يمثل المتحول Multiply القيمة المعادة من التابع الوظيفي. لنوضح أكثر، التابع الوظيفي اسمه Multiply، ويستعمل اسم التابع الوظيفي ليحتوي على القيمة المعادة، ولهذا لا بد من إسناد القيمة التي سيعيدها التابع الوظيفي إلى متحول (يدعى Multiply) ويحمل اسم ذلك التابع.

```
Multiply = Z
```

يستطيع كل من يستدعي التابع الوظيفي Multiply() استخدام القيمة المعادة منه حسب ما توضحه الفقرة التالية:

نص الإجراء cmdCalculate_Click()

يُسنِد نص الإجراء القيمة المعادة من التابع الوظيفي Multiply() إلى الخاصية Text لمربع النص txtResult:

```
Private Sub cmdCalculate_Click()
    txtResult.Text = Str(Multiply(2,3))
End Sub
```

كما تلاحظ، يعتبر استخدام تابع وظيفي أكثر تعقيداً بعض الشيء من استخدام الإجراءات. لكنك ستتعود على استخدام التوابع الوظيفية، وستتمكن من تقدير أهميتها، لأنها تساعد بسهولة على قراءة وتوضيح عمل البرامج. لنأخذ العبارات التالية على سبيل المثال:

```
TtxResult.Text = Str(Multiply(2,3))
```

تقول هذه العبارة ما يلي: " نفّذ التابع الوظيفي Multiply() مع وسيطين هما 2 و3، وتحويل القيمة المعادة من Multiply إلى سلسلة نصية (باستخدام التابع الوظيفي Str())، وإسناد السلسلة الناتجة إلى الخاصية Text لمربع النص". ولهذا، فسوف يحوي مربع النص txtResult على السلسلة "6" داخله (أي 2×3 = 6).

الإجراءات والتوابع الوظيفية والطرائق

حسب ما تقدم في هذا الفصل، فإن الفرق بين الإجراء والتابع الوظيفي أن الإجراء لا يعيد قيمة على عكس التابع الوظيفي. ستصادف عبر الفصول القادمة مصطلحاً جديداً

وهو المصطلح Method (طريقة). تعمل الطريقة بشكل مشابه لعمل الإجراءات والتوابع الوظيفية. وعلى العموم. تنجز الطريقة method نوعاً ما من الوظائفية على كائن خاص ما. فمثلاً تمحو العبارة التالية محتويات النموذج الرسومية، والمدعو frmMyForm:

```
frmMyForm.Cls
```

لاحظ أن اسم الطريقة في العبارة السابقة هو Cls. يمكن أن تبدو الطرائق Methods من وجهة نظر المستخدم كالإجراءات والتوابع الوظيفية ولكن بصيغة غريبة. سنتمرن عبر فصول هذا الكتاب على استخدام الطرائق.

عبارات اتخاذ القرار

تصف الفقرات القادمة كتل البناء البرمجية الأساسية التي تتوافر في لغة فيجول بيسك. ويتولى ما تبقى من الفصل شرح كيفية كتابة برامج تستخدم هذه الكتل.

الإشارة إلى عناصر التحكم في نص البرنامج

كما لاحظنا حتى الآن، يمكننا الإشارة إلى خاصية عنصر تحكم ما، بكتابة اسم عنصر التحكم البرمجي، ثم نقطة (.) ثم اسم الخاصية. فمثلاً، توضح العبارة التالية كيف تشير إلى الخاصية Text لعنصر التحكم txtResult:

```
txtResult.Text
```

كما يمكنك الإشارة أيضاً إلى الخاصية، بإضافة اسم النموذج الذي يتوضع عنصر التحكم فيه. فمثلاً، تشير العبارة التالية إلى الخاصية Text لمربع النص txtResult ضمن النموذج frmMultiply:

```
frmMultiply.txtResult.Text
```

تستطيع في معظم الأحوال تجاهل اسم النموذج الحالي، (لتوفير جهد عملية الإدخال)، وعموماً إذا احتوى البرنامج على أكثر من نموذج، فمن الممكن تحديد اسم النموذج، وسنشاهد أمثلة عبر كتابنا على حالات يتوجب ذكر اسم النموذج فيها.

العبارة If

تعرفنا على كتلة عبارات If.End If في الفصل الثاني. وفي كتلة If.End If اللاحقة، تنفذ العبارات الواقعة بين السطر If والسطر If.End فقط إذا كانت قيمة المتحول A تساوي1:

```
If A = 1 Then
  'ستنفذ العبارات الموجودة هنا
  'إذا كانت قيمة المتحول A = 1
End If
```

تُوضح العبارات التالية شكل عبارات If .Else.End If:

```
If A = 1 Then
  'ستنفذ العبارات الموجودة هنا
  'إذا كانت قيمة المتحول A = 1
Else
  'ستنفذ العبارات الموجودة هنا
  'إذا كانت قيمة المتحول A <> 1
End If
```

العبارة Select Case

يكون استخدام Select Case أحياناً أكثر ملاءمة للاستخدام من If.Else.End If. توضح كتلة العبارات التالية كيف تعمل Select Case:

```
Select Case A
  Case 0
    'ستنفذ العبارات الموجودة هنا
    'إذا كانت قيمة المتحول A = 0
  Case 1
    'ستنفذ العبارات الموجودة هنا
    'إذا كانت قيمة المتحول A = 1
  Case 2
    'ستنفذ العبارات الموجودة هنا
    'إذا كانت قيمة المتحول A = 2
End Select
```

كما ترى، تعمل Select.Case بطريقة مشابهة لعمل عبارة If.

طريقة Do While.Loop

تُستخدم العبارة Do.Loop لتنفيذ العبارات ضمنها، مادام الشرط المحدد محققاً. مثال، تعد حلقة Do.Loop التالية من 1 وحتى 1000:

```
Dim Counter
Counter=1
Do While Counter < 1001

   Counter = Counter + 1
Loop
```

يتم تجهيز المتحول Counter بإسناد القيمة الابتدائية 1إليه، ثم تبدأ بعدها حلقة Do While.Loop.

يتحقق السطر الأول من أن قيمة Counter أصغر من 1001. فإذا كان الأمر كذلك، تنفذ العبارات الواقعة بين السطر Do While والسطر Loop. وفي مثالنا هذا، توجد عبارة واحدة بين هذين السطرين:

```
Counter = Counter +1
```

والتي تزيد قيمة العداد Counter بمقدار 1.

يعود البرنامج بعد ذلك إلى السطر Do While ويتحقق من قيمة العداد Counter، والتي تساوي الآن إلى 2.

ولهذا فسوف يعاد تنفيذ العبارة الواقعة بين السطر Do While والسطر Loop. تستمر المعالجة حتى تصبح قيمة Counter مساوية إلى 1001، وعندها يختل الشرط، ويستأنف تنفيذ البرنامج بدأً من العبارة التي تلي السطر Loop.

طريقة Do.Loop While

العبارات ضمن الحلقة Do While.Loop الموصوفة في الفقرة السابقة قد تنفذ أو لا تنفذ، تبعاً لتحقيق الشرط. فمثلاً العبارات في حلقة Do While.Loop التالية لن تنفذ أبداً:

```
Dim Counter
Counter = 2000
Do While Counter < 1001
   Counter = Counter + 1
```

```
Loop
```

فعندما يتحقق البرنامج من السطر Do While فإنه سيكتشف أن Counter تساوي 2000، ولهذا فلن تنفذ العبارة الواقعة بين السطر Do While والسطر Loop.

يتطلب البرنامج أحياناً الدخول ضمن الحلقة لمرة واحدة على الأقل دون تحقيق أي شرط.

تُستخدم الحلقة Do.Loop While في مثل هذه الحالة:

```
Dim Counter
Counter = 2000
Do
   txtUserArea.Text = Str(Counter)
   Counter = Counter + 1
Loop While Counter < 1001
```

ينفذ البرنامج العبارات الواقعة بين السطر Do والسطر Loop While مهما تكن الأحوال. وبعد ذلك يحدد البرنامج إذا كان الشرط محققاً (أي Counter أصغر من 1001)، يعيد البرنامج تكرار تنفيذ الحلقة إذا تحقق الشرط، وبالتالي يعيد تنفيذ العبارات الواقعة بين السطر Do والسطر Loop While.

أما إذا اختل الشرط (أي Counter ليس أصغر من 1001) فعندها يستأنف التنفيذ من العبارة التي تأتي مباشرة بعد السطر Loop While. تعد الطريقة Do.Loop While التالية من 50 إلى 300:

```
Dim Counter
Counter = 50
Do
   Counter = Counter + 1
Loop While Counter < 301
```

الطريقة For.Next

تُعَدُّ الطريقة For.Next وسيلة أخرى لصنع الحلقات في لغة فيجول بيسك. مثلاً، تَعُدُّ الحلقة التالية من 1 و حتى 100:

```
Dim7 I
For I = 1 to 100 Step 1
   txtMyTextArea.Text = Str(I)
```

```
Next
```

أما للتعداد من 1وحتى 100 وبخطوة (زيادة في كل مرة) قدرها 2، فتستطيع استخدام الحلقة For.Next التالية:

```
Dim I
For I = 1 to 100 Step 2
   txtMyTextArea.Text = Str(I)
Next
```

تعد هذه الحلقة بالشكل التالي: 1،2،3،5، ... 99.

تساوي قيمة Step في الحالة الافتراضية (أي عند تجاهل كتابتها) إلى 1، وبالتالي فالحلقتين التاليتين متماثلتين:

```
Dim I
For I = 1 to 100 Step 1
   txtMyTextArea.Text = Str(I)
Next
```

و الحلقة:

```
Dim I
For I = 1 to 100
   txtMyTextArea.Text = Str(I)
Next
```

العبارة Exit For

تستطيع الخروج من الحلقة For.Next باستخدام عبارة Exit For كما يلي:

```
Dim I
For I = 1 to 1000
   txtResult.Text = Str(I)
   If I = 500 Then
     Exit For
   End If
Next
```

يعد جزء البرنامج هذا بدءاً من الواحد وبزيادة قدرها 1 للمتحول z مع كل تكرار للحلقة.

يتحقق شرط عبارة If الداخلية عندما تصبح قيمة I مساوية إلى 500 (If I = 500) ونتيجة ذلك تنفذ العبارة Exit For التي تنهي بدورها تنفيذ الحلقة For.Next، قبل انتهاء الحلقة.

أي بكلمة أخرى، تتسبب عبارة If بإنهاء الحلقة عند I = 500 (بينما يفترض أن تنتهي الحلقة بدون عبارة If الداخلية عندما تصبح قيمة I مساوية إلى 1000).

العبارة Exit DO

تنتهي الحلقة Do While.Loop باستخدام Exit DO:

```
Dim I
I = 1
Do While I < 10001
   txtResult.Text = Str(I)
   I = I + 2
   If I > 500 Then
     Exit Do
   End If
Loop
```

تعد الحلقة السابقة بدءاً من الواحد وبزيادة قدرها 2. وينهى تنفيذ الحلقة عندما تصبح قيمة I أكبر من 500.

الحلقة اللامنتهية

قد تقع أحياناً في خطأ يشبه ذلك المبين في الحلقة التالية:

```
Dim I
I = 1
Do While I < 10001
   txtResult.Text = Str(I)
   If I > 500 Then
     Exit Do
   End If
Loop
```

كما تلاحظ، نسينا كتابة العبارة التالية:

```
I = I + 2
```

تبقى قيمة Counter ثابتة في الحلقة Do While.Loop السابقة (I=1)، وهذا بسبب نسيان زيادة قيمته. يبقى البرنامج في هذه الحالة ضمن الحلقة إلى اللانهاية، لأن قيمة I دوماً أصغر من 1001 ولكن تكون أبداً أكبر من 500. بل في الواقع تساوي 1 على الدوام.

العبارة With

تمكّنك العبارة With من إسناد القيم إلى عدة خصائص لكائن ما، دون كتابة اسم الكائن لكل خاصية (أي دفعة واحدة).

خذ مثلاً العبارة With التالية التي تسند مجموعة من خصائص عنصر التحكم cmdMyButton:

```
With cmdMyButton
   .Height = 300   .ClientWidth = 4725
   .Caption = "&My Button"
End With
```

تلعب عبارة With نفس دور العبارات التالية:

```
cmdMyButton.Height = 300cmdMyButton.Width = 4725
cmdMyButton.Caption = "&My Button"
```

توفر عبارة With كما نلاحظ وقت الإدخال (زمن كتابة نص البرنامج). فبدلاً من كتابة اسم عنصر التحكم مع كل خاصية. يكفي كتابة اسم عنصر التحكم لمرة واحدة فقط على سطر العبارة With.

برنامج جمع الأعداد

سنصمم الآن برنامجاً يدعى برنامج جمع الأعداد.

يسمح برنامج جمع الأعداد، للمستخدم باختيار رقم ثم يجمع كل الأرقام الصحيحة من 1 وحتى ذلك الرقم. فمثلاً، يُجري برنامج جمع الأعداد، عملية الجمع التالية: عند اختيار الرقم 5:

```
1 + 2 + 3 + 4 + 5 = 15
```

ويظهر الناتج.

التمثيل المرئي لبرنامج جمع الأعداد

سننجز الآن التصميم المرئي لبرنامج جمع الأعداد:

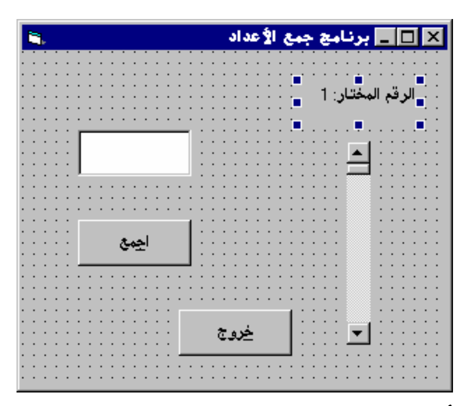
- افتح مشروعاً جديداً من نوع Standard EXE، واحفظ النموذج باسم Sum.frm في الدليل C:\VB5Prg\Ch03 واحفظ المشروع بالاسم Sum.Vbp في ذات الدليل.
- أنشئ النموذج frmSum طبقاً للجدول 3-2.

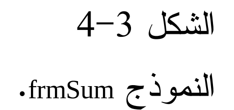
يفترض أن يظهر النموذج المكتمل، كما في الشكل 3-4.

الجدول 3-2. جدول خصائص النموذج frmSum.

الكائن	الخاصية	القيمة
Form	**Name**	**frmSum**
	Caption	برنامج جمع الأعداد
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdSumIt**
	Caption	ا&جمع
	RightToLeft	True
Text Box	**Name**	**txtResult**

	Text	(اجعله فارغاً)
	Alignment	2-Center
	MultiLine	True
	Enabled	False
	RightToLeft	True
Label	**Name**	**lblResult**
	Caption	الرقم المختار: 1
	RightToLeft	True
Vertical Scroll Bar	**Name**	**vsbNum**
	Min	1
	Max	500
	Value	1




الشكل 3-4
النموذج frmSum.

أسندت القيمة False إلى الخاصية Enabled لمربع النص txtResult لمنع المستخدم من الكتابة داخل مربع النص أو تعديله.

إدخال نص برنامج جمع الأعداد

سنكتب الآن نص برنامج جمع الأعداد:

□تحقق بأن العبارة Option Explicit موجودة ضمن قسم التصاريح العامة General Declaration:

```
' يجب التصريح عن كل المتحولات
```

```
Option Explicit
```

▫أدخل النص التالي ضمن الإجراء ()cmdExit_Click للنموذج frmSum:

```
Private Sub cmdExit_Click()
    End
End Sub
```

▫أدخل النص التالي ضمن الإجراء ()cmdSumIt_Click للنموذج frmSum:

```
Private Sub cmdSumIt_Click()
    Dim I
    Dim R
    For I = 1 To vsbNum.Value Step 1
      R = R + I
    Next
    txtResult.Text = Str(R)
End Sub
```

▫أدخل النص التالي ضمن الإجراء ()vsbNum_Change:

```
Private Sub vsbNum_Change()
    lblNum = " الرقم المختار: " + Str(vsbNum.Value)
End Sub
```

▫أدخل النص التالي ضمن الإجراء ()vsbNum_Scroll:

```
Private Sub vsbNum_Scroll()
    vsbNum_Change
End Sub
```

▫احفظ المشروع باختيار **Save Project** من القائمة **File**.

تنفيذ برنامج جمع الأعداد

▫نفّذ برنامج جمع الأعداد.

▫اختر رقماً بالنقر على رمزي الأسهم لشريط التمرير العمودي أو بسحب مؤشر شريط التمرير.

يستجيب البرنامج بإظهار الرقم الذي اخترته.

▫انقر الزر **اجمع**.

يستجيب البرنامج بإظهار ناتج الجمع في مربع النص (انظر الشكل 3-5).



الشكل 3-5
برنامج جمع الأعداد.

فمثلاً، يظهر البرنامج الرقم 15 عند اختيار الرقم 5 بواسطة شريط التمرير العمودي ثم النقر على الزر **اجمع** (أي 1 + 2 + 3 + 4 +5 = 15).

□أنهِ البرنامج بنقر الزر **خروج**.

كيف يعمل برنامج جمع الأعداد

يستخدم البرنامج حلقة For.Next لإنجاز عملية الجمع.

نص الإجراء cmdSumIt_Click()

ينفذ الإجراء cmdSumIt_Click() عند نقر الزر **اجمع**:

```
Private Sub cmdSumIt_Click()
    Dim I
    Dim R
    For I = 1 To vsbNum.Value Step 1
      R = R + I
    Next
    txtResult.Text = Str(R)
End Sub
```

يُصرح الإجراء عن متحولين، وهما I و R، وتحسب الحلقة For.Next المجموع:

```
1 + 2+ 3+ . . . + vsbNum.Value
```

تساوي القيمة الابتدائية للمتحول R إلى صفر، لأن لغة فيجول بيسك، تُسند القيمة الابتدائية صفر إلى المتحولات آلياً، عند التصريح عنها لأول مرة. لكن يفضل بعض المبرمجين استخدام عبارة زائدة، مثل:

```
R = 0
```

قبل عبارة For وذلك زيادة في الإيضاح.

تُحدِّث آخر عبارة في الإجراء، الخاصية Text لمربع النص بمحتويات المتحول R.

نص الإجراء ()vsbNum_Change

يُنفذ هذا الإجراء عند تغيير موضع مؤشر شريط التمرير:

```
Private Sub vsbNum_Change()
    lblNum = " الرقم المختار: " + Str(vsbNum.Value)
End Sub
```

يُحدِّث الإجراء ()vsbNum_Change بإسناد الخاصية Value لشريط التمرير، إلى الخاصية Caption للافتة lblNum، وذلك لتمكين المستخدم من أخذ قراءة الوضعية الحالية لشريط التمرير.

لاحظ أن العبارة:

```
    lblNum = " الرقم المختار: " + Str(vsbNum.Value)
```

هي ذات العبارة:

```
    lblNum.Caption = " الرقم المختار: " + Str(vsbNum.Value)
```

فعند إسناد قيمة إلى عنصر تحكم من نوع لافتة Label بدون ذكر اسم الخاصية، يَفترض فيجول بيسك أنك ترغب بإسناد القيمة إلى الخاصية Caption للافتة، لأن الخاصية Caption هي الخاصية الافتراضية لهذا العنصر (اللافتة).

نمتلك عناصر التحكم الأخرى كحال عنصر اللافتة خصائص افتراضية، فمثلاً الخاصية الافتراضية لعنصر التحكم Text Box (مربع نص) هي الخاصية Text، وهكذا فالعبارة:

```
    txtMyTextBox.Text = " Hello "
```

هي نفس العبارة

```
    txtMyTextBox = " Hello "
```

ملاحظة

تعوّد دائماً على عدم استخدام الخاصية الافتراضية (إلا بشرط ستعرفه في آخر هذه الملاحظة)، لأنك أحياناً قد تخطئ في كتابة اسم الكائن، الذي لم تسند له الخاصية

الافتراضية، ليعتبره فيجول بيسك حينها متحولاً عادياً. مثلاً، افترض وجود مربع نص لديك باسم txtMyText، وأنك تريد إظهار نص معين فيه، (باستخدام ميزة الخاصية الافتراضية)، عندها ستكتب العبارة التالية:

txtMyText = "مرحباً"

ولكنك بدلاً من كتابة العبارة السابقة، كتبت ما يلي خطأً:

txtMText = "مرحباً"

عندها سيعتبر فيجول بيسك أن txtMText عبارة عن متحول عادي، وسيسند له القيمة "مرحباً"، التي لن تظهر في مربع النص txtMyText.

أما إذا كتبت ما يلي:

txtMText.Text = "مرحباً"

عندها سينبهك فيجول بيسك أن txtMText ليس عنصر تحكم، ولا يملك الخاصية Text.

تستطيع حل المشكلة السابقة، بكتابة العبارة Option Explicit في قسم التصاريح العامة، التي تنبهك أن txtMText غير مصرح عنه.

لاحظ مدى أهمية العبارة Option Explicit في برنامجك، وكيف تحل لك الكثير من أخطاء الإدخال، لذلك تعود دائماً على استخدامها.

نص الإجراء vsbNum_Scroll()

ينفذ هذا الإجراء عندما يسحب المستخدم مؤشر شريط التمرير:

```
Private Sub vsbNum_Scroll()
    vsbNum_Change
End Sub
```

ينفذ هذا الإجراء vsbNum_Change()، وهكذا يؤدي تحريك مؤشر شريط التمرير، إلى تنفيذ الإجراء vsbNum_Change() والذي يحدِّث بدوره اللافتة بقيمة الخاصية Value لشريط التمرير.

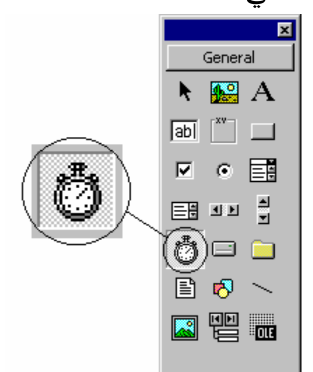
برنامج المؤقِّت

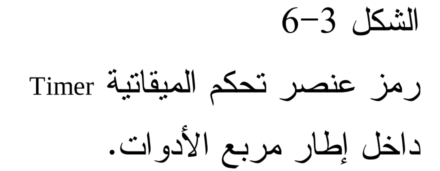
يوضح برنامج المؤقِّت، كيفية استخدام عنصر التحكم Timer.

التمثيل المرئي لبرنامج المؤقِّت

يوضح الشكل 3-6 كيف يبدو عنصر التحكم Timer. ضع مؤشر الفأرة فوق هذا العنصر ليظهر مستطيل أصفر مع الرسالة النصية Timer داخله، وبهذه الطريقة تتأكد من سلامة اختيار العنصر.

لاحظ أن مواقع الرموز ضمن مربع الأدوات قد تختلف عن الموجودة لديك بعض الشيء.





الشكل 3-6

رمز عنصر تحكم الميقاتية Timer

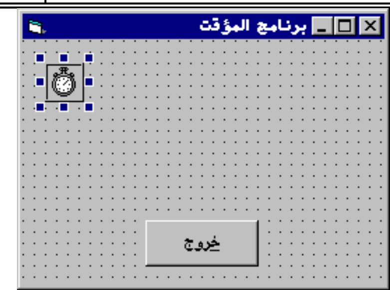
داخل إطار مربع الأدوات.

- أنشئ مشروعاً جديداً من نوع Standard EXE، واحفظ نموذج المشروع بالاسم Timer.frm في الدليل C:\VB5Prg\Ch03 واحفظ ملف المشروع بالاسم Timer.Vbp في ذات الدليل.
- أنشئ النموذج frmTimer طبقاً للجدول 3-3.
- يفترض أن يبدو النموذج عند الكتابة، كما في الشكل 3-7.

الجدول 3-3. جدول خصائص النموذج frmTimer.

الكائن	الخاصية	القيمة
Form	**Name**	**frmTimer**
	Caption	برنامج المؤقت
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج

	RightToLeft	True
Timer	**Name**	**tmrTimer**
	Enabled	True
	Interval	2000



الشكل 3-7

النموذج frmTimer.

إدخال نص برنامج المؤقِّت

سنكتب الآن نص برنامج المؤقِّت:

- تحقق بأن العبارة Option Explicit موجودة ضمن قسم التصاريح العامة :frmTimer

```
' يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء cmdExit_Click() للنموذج frmTimer:

```
Private Sub cmdExit_Click()
    End
End Sub
```

- أدخل النص التالي ضمن الإجراء trmTimer_Timer() للنموذج frmTimer:

```
Private Sub trmTimer_Timer()
    Beep
End Sub
```

- احفظ المشروع باختيار **Save Project** من القائمة **File**.

تنفيذ برنامج المؤقِّت

- نفِّذ برنامج المؤقِّت.

يطلق برنامج المؤقِّت، صوتاً كل ثانيتين (2000 ملي ثانية)، ولا يظهر عنصر التحكم Timer أثناء تنفيذ البرنامج، لكن يظهر فقط أثناء مرحلة التصميم. ستتعرف عبر الكتاب على بعض عناصر التحكم الأخرى التي لا تشاهد أثناء زمن التنفيذ.

- انقر الزر **خروج** لإنهاء البرنامج.

كيف يعمل برنامج المؤقِّت

يستخدم البرنامج عنصر التحكم Timer. والبرنامج ينفذ الإجراء tmrTimer_Timer() آلياً كل 2000 ميلي ثانية (كل ثانيتين).

نص الإجراء tmrTimer_Timer()

تم إسناد القيمة 2000إلى الخاصية Interval للميقاتية tmrTimer أثناء مرحلة التصميم، مما يتسبب بتنفيذ الإجراء tmrTimer_Timer() آلياً كل 2000 ميلي ثانية (ثانيتين).

```
Private Sub trmTimer_Timer()
   Beep
End Sub
```

وهكذا، يعيد الحاسب صوت الرنين كل ثانيتين.

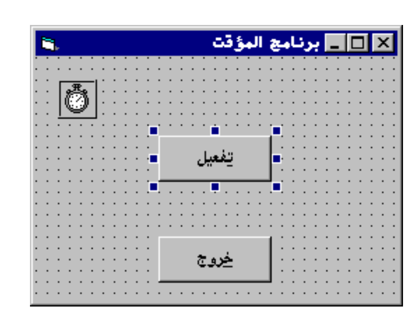
وباعتبار أن رمز الميقاتية Timer لا يظهر عند التنفيذ، وبالتالي فإن موقعه ضمن النموذج ليس هاماً، أي ضعه في أي مكان شئت، بشرط مشاهدته.

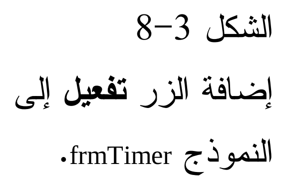
تحسين برنامج المؤقِّت

سنحسن الآن برنامج المؤقِّت:

- أضف زر آخر Command Button، إلى النموذج frmTimer، حسب الشكل 3-8. يفترض أن يحمل زر الأمر هذا الخصائص التالية:

```
Name:          cmdEnableDisable
Caption:       &تفعيل
```




الشكل 3-8

إضافة الزر **تفعيل** إلى

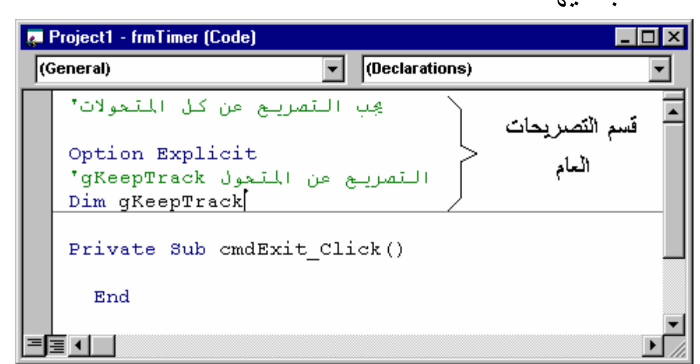
النموذج frmTimer.

❑عدّل قسم التصاريح العامة General Declarations في النموذج frmTimer بحيث يغدو كالتالي:

```
'يجب التصريح عن كل المتحولات
Option Explicit
'gKeepTrack التصريح عن المتحول
Dim gKeepTrack
```

يبين الشكل 3-9 قسم التصاريح العامة General Declarations للنموذج frmTimer.

تستطيع زيادة أو تكبير قسم التصاريح العامة بنقر بداية السطر Option Explicit ثم ضغط المفتاح Enter على لوحة المفاتيح عدة مرات. تتمكن بهذه الطريقة من إضافة عدة أسطر خالية إلى هذا القسم، تستطيع بعد إضافة الأسطر كتابة نص البرنامج المناسب فيها.



الشكل 3-9

قسم التصاريح العامة

للنموذج frmTimer.

❑عدّل الإجراء tmrTimer_Timer().

```
Private Sub Timer_Timer()
    'gKeepTrack أصدر رنيناً إذا كانت قيمة المتحول
    'تساوي الواحد
    If gKeepTrack = 1 Then
      Beep
End If
```

□أضف النص التالي إلى الإجراء cmdEnableDisable:

```
Private Sub cmdEnableDisable_Click()
    If gKeepTrack = 1 Then
      gKeepTrack = 0
      cmdEnableDisable.Caption = "&تفعيل"
    Else
      gKeepTrack = 1
      cmdEnableDisable.Caption = "&تعطيل"
    End If
End Sub
```

□احفظ المشروع بالطريقة المعتادة.

تنفيذ برنامج المؤقِّت

□نفّذ برنامج المؤقِّت. (اضغط المفتاح F5)

لا يصدر البرنامج صوتاً كل ثانيتين.

□انقر الزر **تفعيل**، لتمكين عملية إصدار الصوت.

*يتبدل عنوان الزر **تفعيل** إلى **تعطيل**، ويطلق البرنامج صوت رنين Beep كل ثانيتين (انظر الشكل 3–10).*



الشكل 3–10

الزر cmdEnableDisable

وهو يحمل الآن العنوان **تعطيل**.

□انقر الزر **تعطيل**.

*يعود عنوان الزر إلى **تفعيل**، ويتوقف البرنامج عن إصدار الصوت Beep.*

□كرر عملية النقر على الزر عدة مرات، وبعد انتهائك غادر البرنامج بنقر زر **خروج**.

نص قسم التصاريح العامة للنموذج frmTimer

يتضمن هذا القسم التصريح التالي عن المتحول gKeepTrack:

```
'gKeepTrack التصريح عن المتحول
Dim gKeepTrack
```

السؤال المطروح الآن. ما هو السبب وراء التصريح عن المتحول gKeepTrack هنا ؟!. يفيد التصريح عن المتحولات ضمن هذا القسم إلى جعله مرئياً لكل إجراءات النموذج frmTimer. فمثلاً، لو صرحنا عن المتحول gKeepTrack ضمن الإجراء tmrTimer_Timer() بدلاً من التصريح عنه ضمن قسم التصاريح العامة، فمن الممكن استخدام هذا المتحول ضمن الإجراء tmrTimer_Timer() فقط، ولا يمكن للإجراءات الأخرى الوصول إليه ومعرفة قيمته. ولكن التصريح عنه ضمن قسم التصاريح العامة General Declarations يجعله بمتناول كل الإجراءات والتوابع الوظيفية الموجودة في النموذج frmTimer.

ملاحظة
يدل الحرف الأول من المتحول gKeepTrack هو g، على أن هذا المتحول مصرح عنه في قسم التصاريح العامة General Declarations. ولا يعتبر استخدام هذا الحرف إلزاميا أو من متطلبات لغة فيجول بيسك، إلا أنه أسلوب جيد برمجياً. لهذا سنستخدم الحرف g عبر هذا الكتاب عند التصريح عن المتحولات ضمن قسم التصاريح العامة.

نص الإجراء tmrTimer_Timer()

ينفذ هذا الإجراء كل ثانيتين وذلك بسبب إسناد القيمة 2000 إلى الخاصية Interval لعنصر التحكم trmTimer:

```
Private Sub tmeTimer_Timer()
    'gKeepTrack إذا كانت قيمة المتحول
    'تساوي الواحد
    If gKeepTrack = 1 Then
    Beep
```

```
    End If
End Sub
```

تنفذ العبارة Beep إذا تحقق الشرط If. أي إذا كانت قيمة gKeepTrack تساوي الواحد. لاحظ أننا لم نصرح عن gKeepTrack داخل هذا الإجراء، لكن الإجراء يتمكن من التعرف على المتحول بسبب التصريح عنه ضمن قسم التصاريح العامة كما اتفقنا.

كما ذكرنا أيضاً أن فيجول بيسك يعطي المتحولات القيمة الابتدائية "صفر"، لحظة التصريح عنها، (وبالتالي لا يكون الشرط If هذا محققاً منذ بدء تشغيل البرنامج ولهذا لا يصدر صوتاً كل ثانيتين).

نص الإجراء ()cmdEnableDisable_Click

ينفذ هذا الإجراء عند نقر الزر cmdEnableDisable:

```
Private Sub cmdEnableDisable_Click()
    If gKeepTrack = 1 Then
      gKeepTrack = 0
      cmdEnableDisable.Caption = "&تفعيل"
    Else
      gKeepTrack = 1
      cmdEnableDisable.Caption = "&تعطيل"
    End If
End Sub
```

تنفذ العبارات الواردة ضمن السطرين If و Else إذا كانت قيمة gKeepTrack تساوي "الواحد"، فتصبح قيمة gKeepTrack مساوية الصفر وتبدل الخاصية Caption إلى &تفعيل.

أما إذا كانت قيمة gKeepTrack مساوية للصفر، فستنفذ العبارات الواقعة بين السطر Else و End If، وتصبح قيمة gKeepTrack مساوية إلى "الواحد" وتبدل الخاصية Captionإلى &تعطيل.

أيضاً، يتعرف الإجراء ()cmdEnableDisable_Click على المتحول gKeepTrack وهذا بسبب التصريح عنه ضمن قسم التصاريح العامة.

تعديل برنامج المؤقِّت ثانية

كان الغرض من استخدام المتحول gKeepTrack إيضاح أن المتحول الذي يصرح عنه ضمن القسم General Declarations يصبح مرئياً لكل إجراءات النموذج. تستطيع عموماً كتابة برنامج المؤقِّت بدون استخدام المتحول gKeepTrack. استخدم الخطوات التالية لرؤية كيفية إنجاز ذلك:

- احذف التصريح عن gKeepTrack من قسم التصاريح العامة للنموذج frmTimer، وبالتالي يجب أن يظهر هذا القسم كما يلي:

```
' يجب التصريح عن كل المتحولات
Option Explicit
```

كنا قد طلبنا منك إسناد القيمة True إلى الخاصية Enabled لعنصر التحكم tmrTimer.

- غيّر وضعية الخاصية Enabled إلى False. يؤدي هذا العمل إلى إيقاف تنفيذ الإجراء ()tmrTimer_Timer.
- غيّر الإجراء cmdEnableDisable بحيث يغدو كالتالي:

```
Private Sub cmdEnableDisable_Click()
    If tmrTimer.Enabled = True Then
      tmrTimer.Enabled = False
      cmdEnableDisable.Caption = "&تفعيل"
    Else
      tmrTimer.Enabled = True
      cmdEnableDisable.Caption = "&تعطيل"
    End If
End Sub
```

- غيّر الإجراء ()tmrTimer_Timer بحيث يصبح كالتالي:

```
Private Sub trmTimer_Timer()
    Beep
End Sub
```

- احفظ المشروع.

تنفيذ برنامج المؤقّت

- نفّذ برنامج المؤقِّت وتحقق بأنه يعمل بنفس طريقة استخدام المتحول

.gKeepTrack

□أنهِ البرنامج بنقر الزر **خروج**.

نص الإجراء ()tmrTimer_Timer

ينفذ هذا الإجراء كل 2000 ميلي ثانية (أي كل 2 ثانية) إذا كانت قيمة الخاصية Enabled تساوي True. (طبعاً الفاصل الزمني يساوي 2000 ميلي ثانية، لأننا أسندنا القيمة 2000 إلى الخاصية Interval أثناء طور التصميم).

لن ينفذ الإجراء ()tmrTimer_Timer عند بدء تشغيل البرنامج، وهذا لأننا أسندنا القيمة False إلى الخاصية Enabled، بينما إذا كانت قيمة Enabled تساوي القيمة True فسوف ينفذ الإجراء، ويصدر الحاسب صوتاً Beep كل ثانيتين.

```
Private Sub trmTimer_Timer()
    Beep
End Sub
```

نص الإجراء ()cmdEnableDisable_Click

ينفذ هذا الإجراء عندما ننقر الزر cmdEnableDisable:

```
Private Sub cmdEnableDisable_Click()
   If tmrTimer.Enabled = True Then
     tmrTimer.Enabled = False
     cmdEnableDisable.Caption = "&تفعيل"
   Else
     tmrTimer.Enabled = True
     cmdEnableDisable.Caption = "&تعطيل"
   End If
End Sub
```

تتحقق العبارة If من قيمة الخاصية Enabled للعنصر Timer ؛ فإذا كانت القيمة تساوي True، فسوف تُنفذ العبارات الواقعة بين السطرين If و Else، فتصبح قيمة الخاصية Enabled مساوية إلى False وتتغير الخاصية Caption للزر cmdEnableDisable إلى **تفعيل**.

وعلى العكس، تُنفذ العبارات الواقعة بين السطرين Else و End If إذا كانت قيمة الخاصية Enabled تساوي False، فتصبح قيمتها True وتغير الخاصية Caption للزر cmdEnable Disable إلى **تعطيل**.

تستخدم الكثير من البرامج، تقنية تغيير عنوان الزر (الخاصية Caption)، تبعاً للحالة الراهنة للبرنامج.

ملاحظة

يستخدم برنامج المؤقّت زراً واحداً (الزر cmdEnableDisable)، لإنجاز مهمتي تفعيل وتعطيل (Enable و Disable) عنصر تحكم الميقاتية.

تصبح الخاصية Caption للزر مساوية إلى **تعطيل**، عند تفعيل الميقاتية Timer (وبهذا، يستطيع المستخدم نقر الزر، لتعطيل الميقاتية).

تصبح الخاصية Caption للزر مساوية إلى **تفعيل**، عند تعطيل الميقاتية (ويستطيع المستخدم نقر الزر، لتفعيل الميقاتية).

يُفضل تقليل عدد عناصر التحكم في نموذج البرنامج، قدر الإمكان، واستخدام خاصيتي التفعيل والتعطيل Enable و Disable، وذلك حتى لا نشوش المستخدم. ففي برنامجنا هذا، تمكنا بواسطة زر واحد من إنجاز مهمتين (**تفعيل/تعطيل**).

طبعاً، لا يعني هذا توحيد الأزرار ذات الأدوار المختلفة كلياً. فمثلاً، يجب أن لا نوحّد عمل الزر **خروج** والزر **تفعيل/تعطيل**.

الخلاصة

ناقش هذا الفصل، عبارات اتخاذ القرار في لغة فيجول بيسك وكذلك عبارات الحلقات. تعتبر هذه العبارات أحجار البناء الأساسية للغة البرمجة فيجول بيسك.

كما ركز هذا الفصل على عنصر التحكم Timer (الميقاتيه) ومفهوم مرئية المتحولات وكذلك على التوابع الوظيفية والإجراءات والطرائق.

الفصل الرابع

فـــــأرة

سنتعلم في هذا الفصل كيفية اكتشاف واستخدام الحوادث التي تنجم عن استخدام الفأرة، والتعامل مع النقر على أحد أزرارها أو مع الدمج بين النقر على أحد أزرارها والضغط على لوحة المفاتيح. كما سنتعلم أيضاً ما يتعلق بسحب وإسقاط الكائنات بواسطة الفأرة.

برنامج التحريك

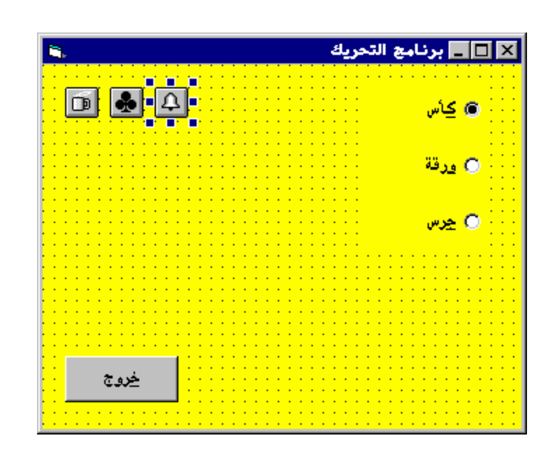
يوضح برنامج التحريك كيفية تصميم برامج تمكن المستخدم من تحريك الكائنات بواسطة الفأرة ضمن إطار البرنامج.

التمثيل المرئي لبرنامج التحريك

سنبدأ بطور التمثيل المرئي للبرنامج:

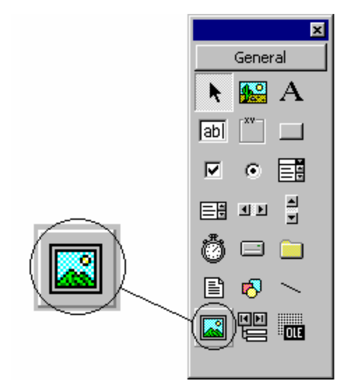
- أنشئ الدليل C:\VB5Prg\Ch04، سنستخدم هذا الدليل لحفظ العمل المنجز فيه.
- أنشئ مشروعاً جديداً من نوع Standard EXE.
- احفظ نموذج المشروع بالاسم Move.Frm في الدليل C:\VB5Prg\Ch04، واحفظ ملف المشروع بالاسم Move.Vbp في نفس الدليل.
- أنشئ النموذج frmMove لبرنامج التحريك طبقاً للجدول 4-1.

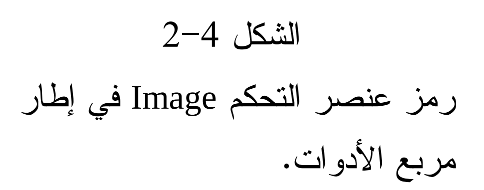
يفترض أن يبدو النموذج عند اكتماله كما في الشكل 4-1.



الشكل 4-1
النموذج frmMove
في مرحلة التصميم.

يبين الشكل 4-2 رمز عنصر التحكم المدعو Image، ولقراءة اسم عنصر التحكم Image ضع مؤشر الفأرة فوق رمزه بدون النقر عليه، فيظهر إطار صغير أصفر يحمل النص Image داخله.




الشكل 4-2
رمز عنصر التحكم Image في إطار مربع الأدوات.

الجدول 4-1. جدول خصائص برنامج التحريك.

الكائن	الخاصية	القيمة
Form	**Name**	**frmMove**
	Caption	برنامج التحريك
	RightToLeft	True

	BackColor	Yellow
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
Option Button	**Name**	**optCub**
	Caption	&كأس
	BackColor	Yellow
	Value	True
	RightToLeft	True
Option Button	**Name**	**optClub**
	Caption	&ورقة
	BackColor	Yellow
	RightToLeft	True
Option Button	**Name**	**optBell**
	Caption	&جرس
	BackColor	Yellow
	RightToLeft	True
Image	**Name**	**imgCup**
	Picture	Cup.Bmp
Image	**Name**	**imgClub**
	Picture	Club.Bmp
Image	**Name**	**imgBell**
	Picture	Bell.Bmp

يطالبك الجدول 4-1 بوضع ثلاثة عناصر تحكم من نوع Image في النموذج frmMove.

تذكر ما يلي لدى استخدام الجدول 4-1:

- لديك حرية تحديد خصائص الخط Font لعناصر التحكم التي تضعها في النموذج.
- يطالبك الجدول بإسناد ملفات متنوعة من نوع BMP إلى الخصائص

Pictures لعناصر التحكم Image. المشكلة أن هذه الملفات قد تكون متوفرة أو غير متوفرة لديك تبعاً لإصدار فيجول بيسك الذي تستخدمه. وهذه الملفات ستجدها في أحد الأدلة الفرعية تحت الدليل الذي تم تنصيب فيجول بيسك فيه. بل ستجدها غالباً تحت الدليل Bitmaps\. فإذا لم تتمكن من العثور عليها، (بسبب أن رزمة فيجول بيسك التي تستخدمها لا توفرها). فبإمكانك استخدام ملفات BMP من أي مصدر آخر. تستطيع مثلاً استخدام برنامج الرسام Paintالذي يأتي مع ويندوز لرسم صور خاصة بك وحفظها كملف من نوع BMP.

إدخال نص برنامج التحريك

سندخل الآن نص برنامج التحريك. ونبدأ كما قمنا بقسم التصاريح العامة General Delarations:

- تحقق من أن قسم التصاريح العامة General Delarations للنموذج frmMove يحوي العبارة التالية:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء ()Form_MouseDown:

```
Private Sub Form_MouseDown(Button As Integer, _
                Shift As Integer, X As Single, Y As Single)
   If optBell.Value = True Then
     imgBell.Move X, Y
   ElseIf optClub.Value = True Then
     imgClub.Move X, Y
   Else
     imgCup.Move X, Y
   End If
End Sub
```

- أدخل النص التالي ضمن الإجراء ()cmdExit_Click:

```
Private Sub cmdExit_Click()
   End
End Sub
```

□احفظ المشروع باختيار **Save Project** من القائمة **File**.

تنفيذ برنامج التحريك

قبل أن نباشر بشرح نص البرنامج الذي كتبناه، دعنا نرى كيف يعمل أولاً:

□نفذ برنامج التحريك.

حسب ما يظهره الشكل 4-3، يظهر البرنامج ثلاث صور، وثلاثة أزرار خيارات معنونة كالتالي **كأس** و **ورقة** و **جرس**.

يؤدي نقر زر الفأرة على نافذة البرنامج، إلى انتقال أحد الصور إلى الموقع الذي تم النقر بالفأرة عنده. تحدَد الصورة المطلوب تحريكها باختيار أحد أزرار الخيارات الثلاثة.

لنفترض مثلاً، أن الخيار يقع عند زر الخيار **الجرس**، يؤدي النقر بالفأرة في أي مكان نافذة البرنامج، إلى انتقال صورة الجرس إلى ذلك المكان (الشكل 4-3).

□أنه البرنامج بنقر الزر **خروج**.



الشكل 4-3

برنامج التحريك

بعد تحريك الجرس.

كيف يعمل برنامج التحريك

ستصادف لدى مناقشتنا برنامج التحريك، مصطلح **إحداثيات النموذج** Form Coordinates. لهذا ستبدأ مناقشتنا لبرنامج التحريك بعد هذا الموضوع.

إحداثيات النموذج Form Coordinates

يمكن تحديد إحداثيات نموذج وفق عدة وحدات قياس بواسطة الخاصية ScaleMode. علماً بأن فيجول بيسك يستخدم وحدة قياس تدعى twip، في الحالة الافتراضية (يوجد 1440 twips في البوصة الواحدة، 1 inch = 1440 twips).

ملاحظة

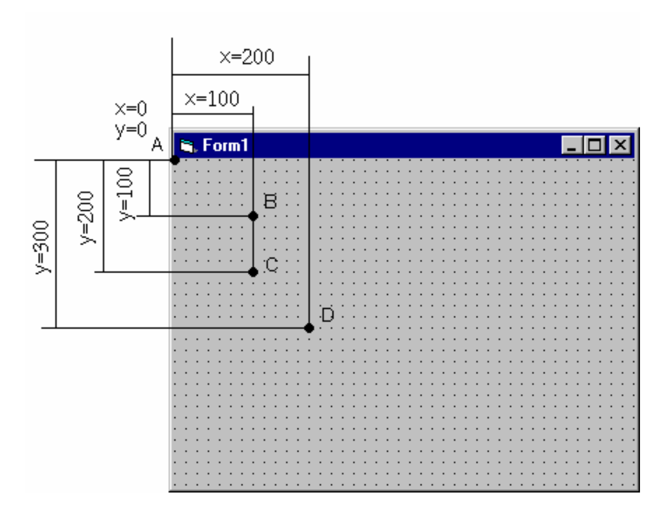
يمكن إسناد أي من الوحدات التالية إلى الخاصية ScaleMode:

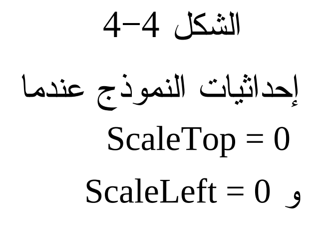
- Twips كل بوصة واحدة تساوي 1440 twips.
- Points كل بوصة واحدة تساوي 72 Points (أي 72 نقطة).
- Pixels يتحدد عدد النقاط الضوئية في كل بوصة تبعاً لدقة تحديد الشاشة Resolution.
- Character يعرف الرّمز Character كمستطيل بعرض 128 Twips وارتفاع 240 twips.
- Inches بوصة.
- Millimeters ميليمترات.
- Centimeters سنتمترات.

يتحدد قيمة مبدأ نظام إحداثيات النموذج بالخاصيتين ScaleTop و ScaleLeft للنموذج. علماً بأن القيمة الافتراضية لهاتين الخاصيتين هي صفر. مما يعني أن الزاوية العليا اليسارية، لمنطقة النموذج تمثل مبدأ الإحداثيات وتتحدد بالإحداثيين (0.0). ويقصد بمنطقة النموذج المنطقة القابلة للاستخدام والتي لا تتضمن الحواف ولا شريط العنوان. (المنطقة التي يمكن وضع عناصر التحكم عليها).

إذاً باختصار، الخاصية ScaleTop للنموذج تساوي صفر، والخاصية ScaleLeft للنموذج تساوي صفر، والزاوية العليا اليسارية للنموذج، تقع عند الإحداثيين (x = 0 و y = 0).

تزداد قيمة x كلما انتقلت باتجاه اليمين، وتزداد قيمة y كلما اتجهت إلى الأسفل (انظر الشكل 4-4).




الشكل 4-4

إحداثيات النموذج عندما
ScaleTop = 0
و ScaleLeft = 0

النقطة A حسب الشكل 4-4، تقع عند الإحداثيين (x = 0 و y = 0). وتقع النقطة B عند الإحداثيين (x = 100 و y = 100)، وتقع C عند (x = 100 و y = 200)، بينما تقع D عند (x = 200 و y = 300).

ملاحظة

يتألف النموذج من شريط عنوان الإطار، وحواف الإطار، والمنطقة المُحَاطَة من شريط العنوان والحواف، والتي تدعى منطقة النموذج.

نص الإجراء cmdExit_Click()

ينفذ هذا الإجراء عند نقر الزر **خروج**. تتسبب عبارة End الموجودة ضمن الإجراء بإنهاء الإجراء:

```
Private Sub cmdExit_Click()
    End
End Sub
```

نص الإجراء Form_MouseDown()

يمكن للفأرة أن تمتلك من زر واحد، وحتى ثلاثة أزرار. يؤدي الضغط على أي من أزرار الفأرة إلى الأسفل، ضمن منطقة النموذج، إلى تنفيذ الإجراء Form_MouseDown() آلياً.

ملاحظة

يقصد بضغط أزرار الفأرة، الضغط عليها إلى الأسفل Down، أما نقر Clicking الفأرة، فيقصد به ضغط أحد الأزرار ثم تحريره. ينفذ الإجراء Form_MouseDown() عند ضغط أحد أزرار الفأرة، (عندما تكون مشيرة الفأرة ضمن منطقة العمل للنموذج). وهكذا فالإجراء ينفذ، حتى قبل تحرير زر الفأرة.

لا ينفذ هذا الإجراء عند ضغط أحد أزرار الفأرة، فوق شريط عنوان النموذج، باعتبار أن منطقة النموذج لا تشمل شريط العنوان ولا الحواف Border. ينفذ الإجراء فقط عند نقر (أو ضغط) زر الفأرة في منطقة خالية (من عناصر التحكم) ضمن منطقة النموذج. فالإجراء Form_MouseDown() مثلاً، لا ينفذ عند ضغط زر الفأرة فوق الزر **خروج**، أو فوق أحد أزرار الخيارات.

وسائط الإجراء Form_MouseDown()

يمتلك الإجراء أربعة وسائط: Button و Shift و X و Y تحتوي هذه الوسائط على معلومات عن حالة الفأرة أثناء النقر عليها:

```
Private Sub Form_MouseDown(Button As Integer, _
                 Shift As Integer, X As Single, Y As Single)
   If optBell.Value = True Then
     imgBell.Move X, Y
   ElseIf optClub.Value = True Then
     imgClub.Move X, Y
   Else
     imgCup.Move X, Y
   End If
End Sub
```

ستجد أن السطر الأول من الإجراء Form_MouseDown() يمتد على سطر واحد في نافذة نص البرنامج، بدلاً من سطرين، كما هي الحال هنا، وذلك طبعاً لأن عرض صفحة الكتاب لا يتسع لكامل السطر.

الوسيط الأول من الإجراء Form_MouseDown() عبارة عن متحول صحيح Integer يدعى Button. تُساعد قيمة هذا الوسيط، بتحديد الزر المضغوط (الأيسر، الأوسط، الأيمن). لا يكترث برنامج التحريك لهذا الوسيط لأنه لا يهتم من هو الزر المضغوط.

الوسيط الثاني من الإجراء أيضاً من النوع الصحيح ويدعى Shift، تشير قيمة هذا الوسيط، إلى حالة مفاتيح التحكم Shift و Ctrl و Alt لحظة ضغط أحد أزرار الفأرة. لعل أفضل اسم لهذا الوسيط هو ShiftCtrlAlt، لكن لا بد من العمل مع الاسم الذي فرضه مصممو فيجول بيسك. لا يكترث برنامج التحريك أيضاً لهذا الوسيط (لا يستخدمه). لأنه لا يبدي أي تصرف خاص تجاه الضغط على أحد هذه المفاتيح أثناء النقر على الفأرة.

الوسيطان الثالث والرابع هما X و Y، ويشيران إلى إحداثيي مؤشر الفأرة لحظة الضغط على زر الفأرة. طبعاً الإحداثيات تؤخذ بدلالة النموذج، لأن الإجراء ينفذ فقط عندما يكون مؤشر الفأرة ضمن منطقة النموذج. لنفترض مثلاً أن الخاصية ScaleMode للنموذج frmMove على الوضعية Twips، وأن قيمة كل من ScaleTop و ScaleLeft تساوي الصفر. إذا أعطى الإجراء Form_MouseDown() تقريراً بأن x = 0 و y = 0 عند النقر على الفأرة في منطقة النموذج فهذا معناه أن الضغط على الفأرة كان عند الزاوية العليا اليسارية. أما إذا كانت x = 10 و y = 20 فهذا يعني أن الضغط كان على بعد 10 twips من يسار النموذج و20 twips تحت قمة النموذج، وهكذا.

عبارة If.Else الواردة ضمن الإجراء Form_MouseDown()

تسهم عبارة If.Else ضمن الإجراء، بتحديد زر الخيار الذي انتقاه المستخدم. فمثلاً إذا كان زر الخيار الحالي هو الزر **جرس** فسوف يتحقق أول جزء من الشرط:

```
If optBell.Value = True Then
```

وستنفذ العبارة التالية:

```
imgBell.Move X, Y
```

التي تستخدم طريقة Move لنقل الصورة إلى الإحداثيين x و y.

طبعاً x و y هما إحداثيي مؤشر الفأرة عندما تم الضغط على زر الفأرة. إذاً باختصار، ستنقل صورة الجرس إلى النقطة ذات الإحداثيين x و y. (تصور أن عنصر التحكم Image محاط بمستطيل، تنقل الطريقة Move عنصر التحكم Image (والصورة المرافقة) بحيث تتوضع الزاوية العليا اليسارية للمستطيل عند الإحداثيين x و y بعد التحريك).

الطريقة Move

تستخدم طريقة Move لتحريك الكائنات المختلفة، مثل النماذج وعناصر التحكم.

استخدمنا العبارة imgCup.Move x,y مثلاً لنقل الكائن المدعو imgCup من موقعه الراهن إلى الموقع الجديد المحدد بالإحداثيين x و y:

```
imgCub.Move X, Y
```

ملاحظة

بعد تنفيذ العبارة التالية:

imgCup.Move x,y

يتحدد الموقع الجديد للزاوية العليا اليسارية للصورة بالإحداثيين x و y.

يؤدي استخدام العبارة التالية إلى توضع مركز الصورة عند الإحداثيين x و y:

imgCup.Move (X - imgCup.Width / 2),(Y - imgCup.Hight / 2)

معلومة هامة عن أزرار الخيارات Option Buttons

أنهينا برنامج التحريك. لكن بقيت نقطة واحدة بخصوص أزرار الخيارات Option Buttons لا بد لك من الإطلاع عليها.

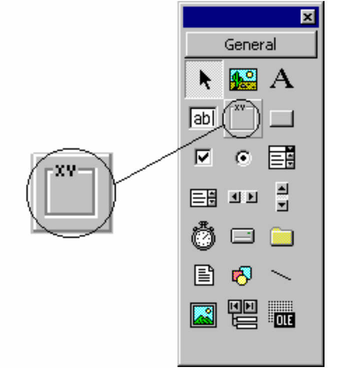
يمكن اختيار زر واحد من أزرار الخيارات التي تنتمي إلى مجموعة واحدة.

ففي مثالنا لدينا ثلاثة أزرار خيارات Option Buttons، ولكن يمكن اختيار واحد فقط من الأزرار الثلاثة، فنستطيع اختيار الجرس أو الكأس أو الورقة ولكن لا تستطيع اختيار الجرس والكأس سوية.

قد تحتاج إلى استخدام أكثر من مجموعة أزرار خيارات واحدة ضمن نفس النموذج. تستطيع إضافة مجموعة ثانية من أزرار الخيارات إلى النموذج وفق ما يلي:

□انقر نقرة مزدوجة على عنصر التحكم Frame في إطار مربع الأدوات (انظر الشكل 4-5 لمشاهدة شكل عنصر التحكم Frame ضمن مربع الأدوات). يؤدي وضع مؤشر الفأرة فوق أي عنصر تحكم ضمن مربع أدوات بدون ضغط أحد أزرار الفأرة، إلى ظهور مستطيل أصفر يحمل اسم عنصر التحكم المرافق. وبهذه الطريقة تستطيع التأكد من اسم عنصر التحكم الصحيح.

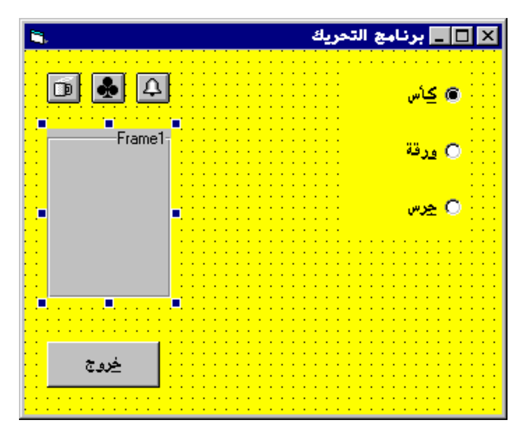
يستجيب فيجول بيسك بوضع عنصر التحكم Frame في النموذج frmMove.



الشكل 4-5

رمز عنصر التحكم Frame

ضمن إطار مربع الأدوات.



الشكل 4-6

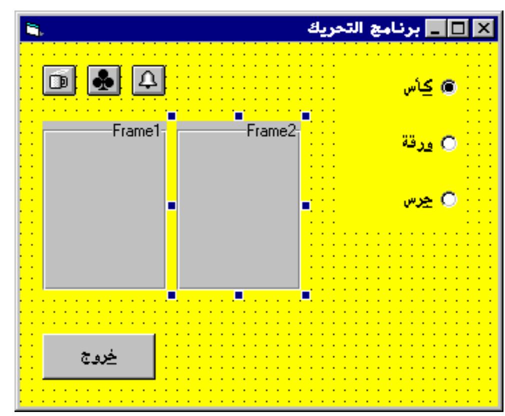
النموذج frmMove بعد إضافة عنصر التحكم Frame إليه (طور التصميم).

□انقر نقرة مزدوجة على رمز عنصر التحكم Frame مجدداً ضمن إطار مربع الأدوات.

يستجيب فيجول بيسك بوضع عنصر تحكم Frame آخر في النموذج frmMove.

□انقل وكبر عنصر التحكم Frame الثاني الذي وضعناه للتو في النموذج frmMove.

يفترض أن يبدو النموذج الآن حسب ما هو مبين في الشكل 4-7.



الشكل 4-7

النموذج frmMove بعد إضافة

عنصري تحكم Frame إليه.

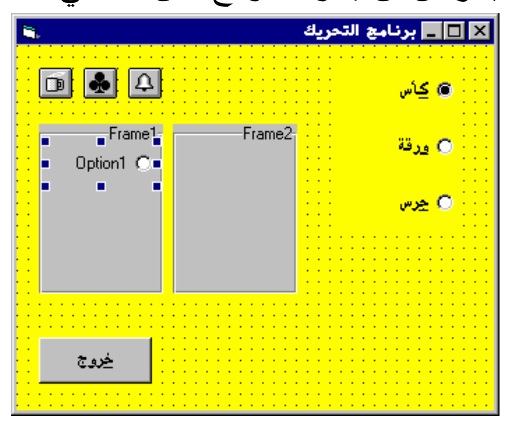
□ضع الآن زري خيار في أول إطار Frame أضفناه إلى النموذج:

□انقر (لا تنقر نقرة مزدوجة) على رمز زر الخيار في إطار مربع الأدوات.

□ضع مؤشر الفأرة ضمن عنصر التحكم Frame الأول واسحب الفأرة.

يستجيب فيجول بيسك بوضع زر خيار في عنصر التحكم Frame.

يفترض أن يبدو النموذج الآن كما في الشكل 4-8.



الشكل 4-8

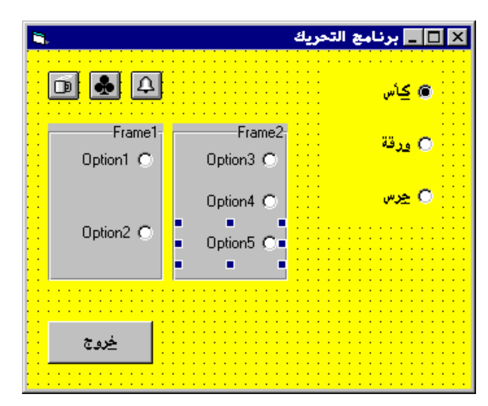
النموذج frmMove بعد إضافة

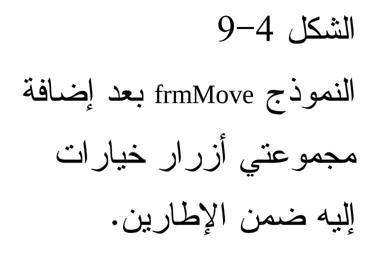
زر خيار إلى عنصر التحكم

Frame الأول.

□كرر خطوات إضافة زر خيار إلى الإطارين. تذكر عدم النقر المزدوج على رمز زر الخيار في إطار مربع الأدوات. ضع زر خيار ثاني في الإطار الأول، وضع ثلاثة أزرار خيارات في الإطار الثاني.

يفترض أن يبدو النموذج كما في الشكل 4-9.




الشكل 4-9

النموذج frmMove بعد إضافة مجموعتي أزرار خيارات إليه ضمن الإطارين.

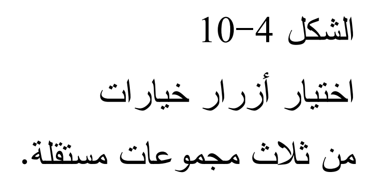
□احفظ المشروع باختيار **Save Project** من القائمة **File**.

□نفّذ برنامج التحريك.

النقطة الجديرة بالملاحظة أن النموذج frmMove يمتلك الآن ثلاثة مجموعات مستقلة من أزرار الخيارات Option Button. ولا يؤثر اختيار زر من مجموعة أولى على أي من المجموعات المتبقية، فاختيار **كأس** مثلاً لا يؤثر على المجموعتين الأخريتين. كذلك اختيار Option1 و Option3 لا يؤثر على المجموعة الأولى. تستطيع التحقق من ذلك وفق ما يلي:

□ انتق Option1 و Option3 و**كأس** ثم انظر إلى الشكل 4-10.




الشكل 4-10

اختيار أزرار خيارات من ثلاث مجموعات مستقلة.

□بدّل الآن الخيارات في كل مجموعة من مجموعات أزرار الخيارات وراقب النتائج ؟

□أنهِ البرنامج الآن بنقر الزر **خروج**.

انتهينا الآن من التمرن على برنامج التحريك. تستطيع إن شئت حذف الإطارين Frame1 و Frame2 وأزرار الخيارات Option1 و Option2 و Option3 و Option4 و Option5.

ولحذف أي من عناصر التحكم هذه، اختر العنصر بنقره مرة واحدة ثم ضغط المفتاح Delete على لوحة المفاتيح. وتستطيع إن شئت حذف كل زر من أزرار الخيارات ضمن الإطار ثم حذف الإطار، أو تستطيع حذف الإطار برمته مما سيتسبب بحذف كل عناصر التحكم الموجودة فيه.

برنامج الرسم

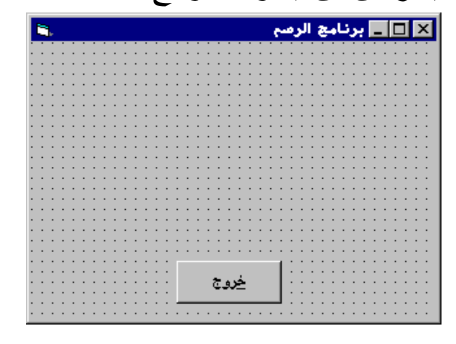
يوضح برنامج الرسم، كيف يمكن لبرنامج ما، استخدام حوادث الفأرة لإنشاء برنامج رسم بسيط.

التمثيل المرئي لبرنامج الرسم

سنبدأ كالعادة بطور التمثيل المرئي للبرنامج:

- أنشئ مشروعاً جديداً من نوع Standard EXE.
- احفظ نموذج المشروع بالاسم Draw.Frm في الدليل C:\VB5Prg\Ch04، واحفظ ملف المشروع بالاسم Draw.Vbp في نفس الدليل.
- أنشئ النموذج frmDraw لبرنامج الرسم طبقاً للجدول 4 -2.

يفترض أن يبدو النموذج عند اكتماله كما في الشكل 4 -11.



الشكل 4 -11

النموذج frmDraw

(طور التصميم).

الجدول 4-2. جدول خصائص النموذج frmDraw.

القيمة	الخاصية	الكائن
FrmDraw	**Name**	**Form**
برنامج الرسم	Caption	
True	RightToLeft	
cmdExit	**Name**	**CommandButton**
&خروج	Caption	
True	RightToLeft	

إدخال نص برنامج الرسم

- تحقق من وجود العبارة Option Explicit ضمن قسم التصاريح العامة.

```
Option Explicit
```

- أدخل النص التالي ضمن الإجراء ()Form_MouseDown:

```
Private Sub Form_MouseDown(Button As Integer, _
                   Shift As Integer,X As Single, Y As Single)
   frmDraw.CurrentX = X
   frmDraw.CurrentY = Y
End Sub
```

- أدخل النص التالي ضمن الإجراء ()Form_MouseMove:

```
Private Sub Form_MouseMove(Button As Integer, _
                   Shift As Integer,X As Single, Y As Single)
   If Button = 1 Then
     Line (frmDraw.CurrentX, frmDraw.CurrentY)- _
          (X, Y), QBColor(0)
   End If
End Sub
```

- أدخل النص التالي ضمن الإجراء ()cmdExit_ Click:

```
Private Sub cmdExit_Click()
   End
End Sub
```

- احفظ العمل المنجز باختيار **Save Project** من القائمة **File**.

تنفيذ برنامج الرسم

▫نفّذ برنامج الرسم.

يظهر الإطار الفارغ المبين في الشكل 4-12.

▫اضغط زر الفأرة وحركها أثناء ذلك. لترسم خطوط تبعاً لحركة الفأرة.

يسبب تحرير زر الفأرة بإيقاف الرسم. باختصار يمكنك الرسم بنفس الطريقة المتبعة في برنامج الرسام Paint الذي يأتي مع ويندوز. يبين الشكل 4-13شكلاً يمكن رسمه ببرنامجنا هذا.



الشكل 4-12

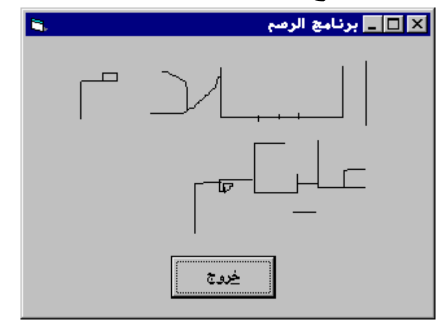
إظهار برنامج الرسم حال تشغيل البرنامج.



الشكل 4-13

الرسم بواسطة برنامج الرسم.

كما يوضح الشكل 4-14 كيف تستطيع الكتابة بواسطة برنامج الرسم.



الشكل 4-14

الكتابة بواسطة برنامج الرسم.

▫أنهِ البرنامج بعد انتهائك من الرسم بنقر الزر **خروج**.

كيف يعمل برنامج الرسم

يستخدم برنامج الرسم مضمونين متعلقين بالرسوم في فيجول بيسك: طريقة Line والخاصيتين CurrentX و CurrentY. ولفهم برنامج الرسم يلزمنا أولاً فهم هذين المضمونين.

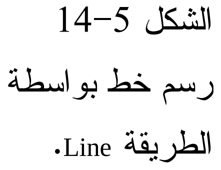
الطريقة Line

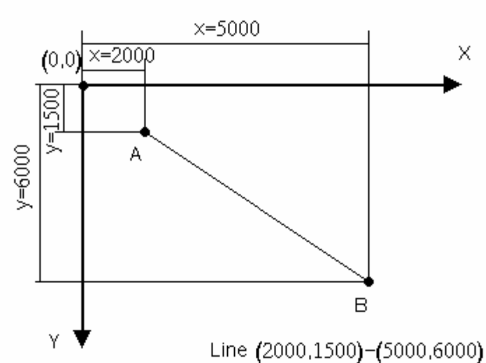
تستخدم طريقة Line لرسم خط مستقيم، من نقطة إلى نقطة، ضمن نموذج. فمثلاً، تستخدم العبارة التالية:

```
Line (2000, 1500) - (5000, 6000)
```

لرسم خط من x =2000 ، y =1500 إلى x =5000 ، y =6000 .

يبين الشكل 4-15 الخط المرسوم نتيجة تنفيذ هذه العبارة. يبدأ الخط على بعد 2000 twips من الجانب الأيسر للنموذج و 1500 twips من قمة النموذج (أي من النقطة A في الشكل 4-15)، وينتهي على بعد 5000 twips من الجانب الأيسر للنموذج 6000 twips من قمة النموذج (النقطة B في الشكل 4-15).


الشكل 14-5
رسم خط بواسطة
الطريقة Line.



يمكن استخدام التابع الوظيفي QBColor() لتحديد اللون الذي سيرسم الخط به. يبين الجدول 4-3 الأرقام التي يستخدمها QBColor().

الجدول 4-3. الأرقام المستخدمة في التابع الوظيفي QBColor().

Color	Number

Black	0
Blue	1
Green	2
Cyan	3
Red	4
Magenta	5
Yellow	6
White	7
Gray	8
Light Blue	9
Light Green	10
Light cyan	11
Light Red	12
Light Magenta	13
Light Yellow	14
Bright White	15

فمثلاً لرسم نفس الخط المبين بالشكل 4−15 بلون أسود، استخدم العبارة:

```
Line (2000, 1500) - (5000, 6000),QBColor(0)
```

ولرسم نفس الخط بلون أحمر، استخدم العبارة التالية:

```
Line (2000, 1500) - (5000, 6000),QBColor(4)
```

الخاصيتين CurrentX و CurrentY للنموذج

لن تجد لدى تفحصك خصائص النموذج، الخاصتين CurrentX و CurrentY لأن فيجول بيسك لا يمكّنك من إسناد قيمة إليهما في مرحلة التصميم. فقيمة هاتين الخاصتين يمكن تغييرها فقط أثناء زمن التنفيذ (أي ضمن نص البرنامج).

تُحَدَّثُ قيمة الخاصتين CurrentX و CurrentY آلياً من قبل البرنامج بعد استخدام شتى الطرق الرسومية. فمثلاً، يسند البرنامج بعد رسم خط بطريقة Line إحداثيات نقطة نهاية الخط إلى الخاصتين CurrentX و CurrentY. وهكذا فبعد تنفيذ العبارة التالية:

```
Line (2000, 1500) - (5000, 6000),QBColor(0)
```

تصبح قيمة CurrentX تساوي5000 وقيمة CurrentY تساوي 6000.

نص الإجراء Form_MouseDown()

ينفذ الإجراء Form_MouseDown() آلياً، عند ضغط زر الفأرة ضمن منطقة النموذج:

```
Private Sub Form_MouseDown(Button As Integer, _
                Shift As Integer, X As Single, Y As Single)
   frmDraw.CurrentX = X
   frmDraw.CurrentY = Y
End Sub
```

يسند هذا الإجراء الوسيطين X و Y إلى الخاصتين CurrentX و CurrentY للنموذج frmDraw على التوالي. وهكذا فبعد ضغط زر الفأرة، يشير الإحداثيان CurrentX و CurrentY إلى المكان الذي ضُغط فيه زر الفأرة.

نص الإجراء Form_MouseMove()

ينفذ هذا الإجراء آلياً عند تحريك الفأرة ضمن منطقة النموذج. يحمل الوسيطان X و Y لهذا الإجراء نفس معنى X و Y في الإجراء Form_MouseMove() (أي بمعنى أن X و Y يمثلان إحداثيي مؤشر الفأرة).

```
Private Sub Form_MouseMove(Button As Integer, _
                  Shift As Integer, X As Single, Y As Single)
   If Button = 1 Then
     Line (frmDraw.CurrentX, frmDraw.CurrentY)- _
          (X, Y), QBColor(0)
   End If
End Sub
```

يختبر الإجراء قيمة Button للتحقق من أنه يجري الآن الضغط على الزر الأيسر للفأرة. فإذا كانت قيمة Button تساوي "1" فهذا معناه أنه يتم الضغط على الزر الأيسر للفأرة، وبالتالي تنفذ العبارة التالية:

```
Line (frmDraw.CurrentX, frmDraw.CurrentY)-(X,Y), QBColor(0)
```

ترسم هذه العبارة خطاً بلون أسود بدءاً من الموقع المحدد بالخاصتين CurrentX و CurrentY إلى الموقع الراهن للفأرة.

تذكر دائماً أن البرنامج يُحَدَّثُ قيمة الخاصتين CurrentX و CurrentY بعد تنفيذ طريقة Line آلياً بإحداثيات نقطة نهاية الخط. مما يعني أنّ التنفيذ التالي للإجراء Form_MouseMove() سيتسبب برسم خط جديد بدءاً من نقطة نهاية الخط السابق.

ينفذ البرنامج الإجراء Form_MouseMove() عند تحريك الفأرة، ويرسم خطاً جديداً بدءاً من نهاية الخط السابق وحتى الموقع الحالي للفأرة في حال ضغط المستخدم على الزر الأيسر للفأرة.

لاحظ أن هذا الإجراء ينفذ مجدداً ومجدداً كلما حركت الفأرة. وإذا تم الضغط أثناء التحريك على زر الفأرة الأيسر فسيرسم خط مع كل تنفيذ للإجراء Form_MouseMove(). والخط يرسم بدءاً من نقطة نهاية الخط السابق وحتى الموقع الحالي لمؤشر الفأرة.

نص الإجراء cmdExit_Click()

```
Private Sub cmdExit_Click()
    End
End Sub
```

الخاصية AutoRedraw

هناك خلل في برنامج الرسم. اتبع الخطوات التالية للتعرف على المشكلة:

- نفّذ برنامج الرسم وارسم عدة خطوط بواسطته.
- صغِّر إطار البرنامج. (انقر رمز إشارة الناقص الذي يظهر عند الزاوية العليا من الإطار لتصغير النافذة).

يستجيب برنامج الرسم بإظهار نفسه كرمز (Icons) على شريط مهام ويندوز.

- استرجع إطار برنامج الرسم. (انقر رمز البرنامج الموجود حالياً على شريط مهام ويندوز Windows).

كما تشاهد، اختفت الخطوط التي رسمتها.

تظهر نفس المشكلة عند تغطية جزء من إطار برنامج الرسم بإطار آخر. تحل هذه المشكلة ببساطة، بإسناد القيمة True إلى الخاصية AutoRedraw للنموذج frmRedraw في طور التصميم.

□أسند القيمة True إلى الخاصية AutoRedraw للنموذج frmRedraw.

□احفظ المشروع باختيار **Save Project** من القائمة **File** لفيجول بيسك.

يعيد فيجول بيسك رسم الإطار عندما تدعو الحاجة، عندما تكون قيمة الخاصية AutoRedraw تساوي True (حسب ما يتوضح من اسم الخاصية نفسها). لنشاهد ذلك على أرض الواقع.

□نفّذ برنامج الرسم وارسم بضعة خطوط.

□ صغر إطار البرنامج. (بنقر رمز التصغير الموجود عند الزاوية العليا اليمنى من الإطار).

يستجيب البرنامج بإظهار نفسه كرمز على شريط مهام ويندوز.

□استرجع إطار برنامج الرسم. (انقر رمز البرنامج الموجود الآن على شريط مهام ويندوز).

كما تشاهد، تظهر الخطوط التي كانت مرسومة.

إذاً تعلمنا أن إسناد قيمة True إلى الخاصية AutoRedraw يمّكن البرنامج من إعادة رسم محتويات الإطار عندما تدعو الحاجة إلى ذلك.

حاول ألا تستخدم هذه الخاصية، رغم أنها قد تبدو ملاءمة للاستخدام ضمن البرنامج. قد تستغرب!!. فهذه الخاصية تتسبب بانخفاض أداء البرنامج. سنتعلم طرائق أخرى من هذا الكتاب أكثر ملاءمة لإعادة إنعاش الإطار.

برنامج الفأرة الراسمة

ينبغي أن لا يُشغل البرنامج المكتوب بلغة فيجول بيسك نفسه على الدوام بتنفيذ الإجراء Form_MouseMove() عند تحريك الفأرة، لأن ذلك يمنعه من تنفيذ أي برنامج آخر أثناء تحريك الفأرة. يتفحص برنامجنا هذا حالة الفأرة خلال فترات زمنية ثابتة

فقط مما يسمح له بتنفيذ المهام الأخرى عند تحريك الفأرة. فإذا اكتشف البرنامج أن الفأرة حركت منذ آخر اختبار، فسوف ينفذ الإجراء ()Form_MouseMove.

التمثيل المرئي لبرنامج الفأرة الراسمة

سنبدأ كعادتنا بطور التمثيل المرئي لنموذج البرنامج:

- أنشئ مشروعاً جديداً من نوع Standard EXE.
- احفظ نموذج المشروع بالاسم HowOften.Frm في الدليل C:\VB5Prg\Ch04، واحفظ ملف المشروع بالاسم HowOften.Vbp في نفس الدليل.
- أنشئ النموذج frmHowOften طبقاً للجدول 4-4.

يفترض أن يبدو النموذج عند اكتماله كما في الشكل 4-16.



الشكل 4-16

النموذج frmHowOften

(في طور التصميم).

الجدول 4-4. جدول خصائص النموذج frmHowOften.

الكائن	الخاصية	القيمة
Form	**Name**	**frmHowOften**
	Caption	برنامج الفأرة الراسمة
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True

إدخال نص برنامج الفأرة الراسمة

- كالعادة، تحقق من وجود العبارة Option Explicit ضمن قسم التصاريح العامة General Declaration للنموذج frmHowOften. (أي اكتبها إذا لم تكن موجودة).

```
' يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء ()Form_MouseMove:

```
Private Sub Form_MouseMove(Button As Integer, _
                 Shift As Integer, X As Single, Y As Single)
    Circle (X, Y), 40
End Sub
```

- أدخل النص التالي ضمن الإجراء ()cmdExit_ Click:

```
Private Sub cmdExit_Click()
    End
End Sub
```

تنفيذ برنامج الفأرة الراسمة

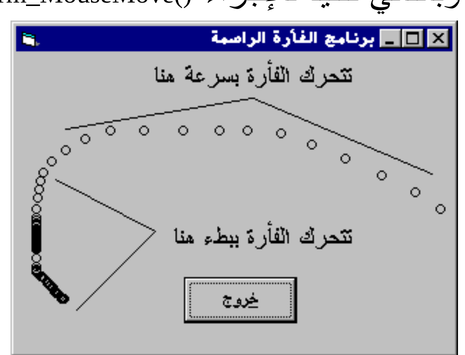
لنشاهد أثر ما كتبناه:

- تنفيذ برنامج الفأرة الراسمة.
- ضع مؤشر الفأرة في منطقة النموذج، ومن ثم حرك الفأرة.

يرسم الإجراء Form_MouseMove() مع تحريك الفأرة دوائر صغيرة عند الموقع الحالي للفأرة، وعموماً ستلاحظ أن الإجراء Form_MouseMove() لن ينفذ مع كل حركة للفأرة. (انظر الشكل 4-17).

يرسم الإجراء Form_MouseMove() قطاراً من الدوائر لدى تحريك الفأرة، وتزداد كثافة الدوائر المرسومة عند تحريك الفأرة ببطئ. ويقل عدد الدوائر المرسومة عند تحريك الفأرة بسرعة.

تذكر أن كل دائرة مرسومة هي دلالة على وقوع حادثة تحريك فأرة MouseMove. وبالتالي تنفيذ للإجراء Form_MouseMove().



الشكل 4-17
تحريك الفأرة في منطقة نموذج برنامج الفأرة الراسمة.

كيف يعمل برنامج الفأرة الراسمة

يستخدم برنامج الفأرة الراسمة الإجراء Form_MouseMove() لإنجاز العمل.

نص الإجراء Form_MouseMove()

ينفذ الإجراء Form_MouseMove() عندما يكتشف البرنامج أنه تم تحريك الفأرة بعد آخر عملية تحقق:

```
Private Sub Form_MouseMove(Button As Integer, _
                 Shift As Integer, X As Single, Y As Single)
    Circle (X, Y), 40
End Sub
```

يرسم هذا الإجراء عند الموقع الحالي لمؤشر الفأرة باستخدام العبارة:

```
Circle (x, y) , 40
```

إلى رسم دائرة بقطر يساوي 40 وحدة قياس Unit (أي حسب وحدة القياس المستخدمة وباعتبار أن الخاصية ScaleMode للنموذج تشير إلى Twip، فهذا يعني أن الطريقة Circle ستستخدم وحدة القياس Twip (القطر يقاس بالـ twips ويقاس مركز الإحداثيات بالـ twips).

نص الإجراء cmdExit_Click()

ينفذ هذا الإجراء عند نقر الزر **خروج** وينهي بدوره تنفيذ البرنامج:

```
Private Sub cmdExit_Click()
    End
End Sub
```

برنامج أزرار الفأرة

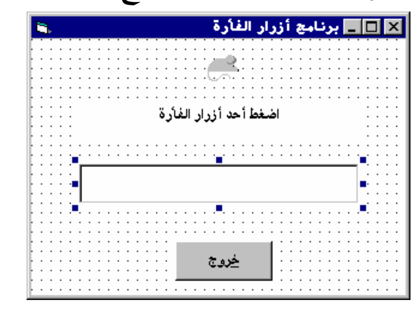
سنكتب الآن برنامجاً يدعى برنامج أزرار الفأرة، يستخدم الوسيط Button لكل من الإجراءيين Form_MouseDown() و Form_MoveUp() لتحديد زر الفأرة الذي تم ضغطه أو تحريره.

التمثيل المرئي لبرنامج أزرار الفأرة

سنبني الآن نموذج برنامج أزرار الفأرة:

- أنشئ مشروعاً جديداً من نوع Standard EXE.
- احفظ نموذج المشروع بالاسم Button.Frm في الدليل C:\VB5Prg\Ch04 واحفظ ملف المشروع بالاسم Button.Vbp في نفس الدليل.
- أنشئ النموذج frmButton طبقاً للجدول 4-5.

يفترض أن يبدو النموذج عند اكتماله كما في الشكل 4-18.



الشكل 4-18
النموذج frmButton
(في طور التصميم).

الجدول 4–5. جدول خصائص النموذج frmButton.

الكائن	الخاصية	القيمة
Form	**Name**	**frmButton**
	Caption	برنامج أزرار الفأرة
	BackColor	White
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
Text Box	**Name**	**txtResult**
	Text	(اجعله فارغاً)
	Alignment	2-Center
	MultiLine	True
	Enabled	False
	RightToLeft	True
Label	**Name**	**lblInstruction**
	Caption	اضغط أحد أزرار الفأرة
	Alignment	2-Center
	BackColor	White
	RightToLeft	True
Image	**Name**	**imgMouse**
	Picture	Mouse04.Ico
	Stretch	False

يطالبك الجدول 4–5 بإسناد اسم الملف Mouse04.Ico إلى الخاصية Picture لعنصر التحكم imgMouse. يقع هذا الملف في حال توفره ضمن الدليل الفرعي \Icons\Computer\ تحت الدليل الذي تم تنصيب لغة فيجول بيسك فيه. يمكنك طبعاً إسناد أي ملف آخر من نوع Ico إلى الخاصية Picture. ولا سيما إذا لم تجد الملف المذكور. فالغرض

الوحيد من استخدام هذا الملف في برنامجنا أنه يتلاءم مع سياق العمل لأنه يمثل شكل الفأرة.

إدخال نص برنامج أزرار الفأرة

- كالعادة، تحقق (أو أدخل) من وجود العبارة التالية: Option Explicit ضمن قسم التصاريح العامة General Declaration للنموذج frmButton:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء Form_MouseDown():

```
Private Sub Form_MouseDown(Button As Integer, _
              Shift As Integer, X As Single, Y As Single)
   If Button = 1 Then
     txtResult.Text = "زر الفأرة اليساري مضغوط الآن"
   End If
   If Button = 2 Then
     txtResult.Text = "زر الفأرة اليميني مضغوط الآن"
   End If
   If Button = 4 Then
     txtResult.Text = "زر الفأرة الوسطي مضغوط الآن"
   End If
End Sub
```

- أدخل النص التالي ضمن الإجراء Form_MouseUp():

```
Private Sub Form_MouseUp(Button As Integer, _
               Shift As Integer, X As Single, Y As Single)
   txtResult = ""
End Sub
```

- أدخل النص التالي ضمن الإجراء cmdExit_ Click():

```
Private Sub cmdExit_Click()
   End
End Sub
```

- احفظ المشروع باختيار **Save Project** من القائمة **File**.

تنفيذ برنامج أزرار الفأرة

- نفّذ برنامج أزرار الفأرة.

يظهر الإطار المبين في الشكل 4–19 عند بدء تشغيل برنامج أزرار الفأرة.



الشكل 4–19

إطار برنامج أزرار الفأرة.

يُظهر مربع النص اسم الزر المضغوط عند ضغط أحد أزرار الفأرة في منطقة خالية ضمن النموذج. يوضح الشكل 4–20 محتوى مربع النص نتيجة ضغط الزر الأيمن للفأرة. ويبين الشكل 4–21 محتوى مربع النص عند ضغط الزر الأيسر للفأرة. يُظهر برنامج أزرار الفأرة حالة زر الفأرة فقط عند الضغط على أحد أزرار الفأرة في منطقة خالية من النموذج.



الشكل 4–20

برنامج أزرار الفأرة عند

الضغط على الزر الأيمن للفأرة.



الشكل 4–21

برنامج أزرار الفأرة عند الضغط

على الزر الأيسر للفأرة.

ملاحظة

تدعى المنطقة غير المغطاة بعنصر تحكم فعال Enabled ضمن النموذج، بمنطقة خالية. فمثلاً الخاصية Enabled للافتة lblInstruction تساوي True، ولهذا لا تعتبر المنطقة التي تشغلها هذه اللافتة منطقة خالية.
ومن جهة أخرى، تساوي الخاصية Enabled لمربع النص txtResult إلى False وبالتالي فالمنطقة التي يشغلها مربع النص هذا عبارة عن منطقة خالية ضمن النموذج.
لا يؤدي النقر على اللافتة lblInstruction إلى تنفيذ الإجراء Form_MouseDown()، لأن منطقة هذه اللافتة كما اتفقنا لا تمثل منطقة خالية.
وعلى العكس تعتبر منطقة مربع النص txtResult منطقة خالية ويؤدي النقر عليها إلى تنفيذ الإجراء Form_MouseDown().

لا يستجيب برنامج أزرار الفأرة عند النقر على الزر الأوسط للفأرة ما لم يتم تعريف الفأرة ضمن ويندوز بأنها ذات ثلاثة أزرار. (طبعاً في حال الفأرة ذات ثلاثة أزرار).

كيف يعمل برنامج أزرار الفأرة

يستجيب برنامج أزرار الفأرة لحادثة MouseDown (الضغط على زر فأرة) بتنفيذ الإجراء Form_MouseDown() ولحادثة تحرير الفأرة MouseUp بتنفيذ الإجراء Form_MouseDown().

نص الإجراء Form_MouseDown()

ينفذ هذا الإجراء تلقائياً بضغط أحد أزرار الفأرة في منطقة خالية ضمن النموذج.
مهمة الإجراء تحديد الزر الذي تم ضغطه بفحص قيمة الوسيط Button.
عندما تكون قيمة الوسيط Button تساوي 1 فهذا معناه أنه تم الضغط على الزر الأيسر، وتعني القيمة 2 الضغط على الزر الأيمن، والقيمة 4 تشير إلى الضغط على الزر الأوسط:

```
Sub Form_MouseDown(Button As Integer,Shift As Integer,_
                   X As Single, Y As Single)
   If Button = 1 Then
     txtResult.Text = "زر الفأرة اليساري مضغوط الآن"
```

```
    End If
    If Button = 2 Then
      txtResult.Text = "زر الفأرة اليميني مضغوط الآن"
    End If
    If Button = 4 Then
      txtResult.Text = "زر الفأرة الوسطي مضغوط الآن"
    End If
End Sub
```

نص الإجراء Form_MouseUp()

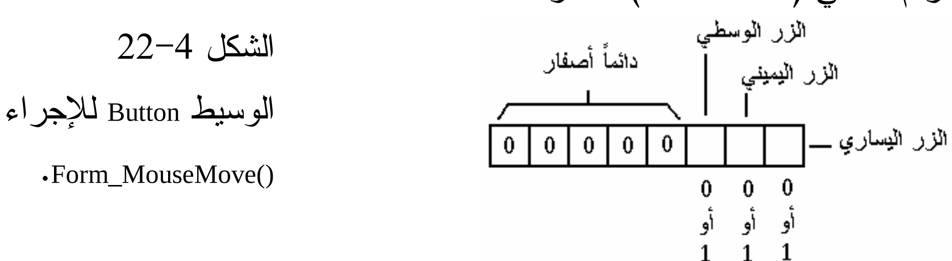
ينفذ هذا الإجراء تلقائياً عند تحرير زر الفأرة. يمحو نص هذا الإجراء محتويات مربع النص:

```
Private Sub Form_MouseUp(Button As Integer, _
                 Shift As Integer, X As Single, Y As Single)
    txtResult = ""
End Sub
```

الوسيط Button في الإجراء Form_MouseMove()

يحدد الوسيط Button في إجراءات الفأرة الزر المضغوط لحظة وقوع الحادثة. وحسب ما ذكرنا فإن قيمة هذا الوسيط ضمن الإجراء Form_MouseDown() يمكن أن تساوي 1 أو 2 أو 4 ولا يمكن أن تتخذ أي قيمة أخرى مما يعني أنك لا تستطيع استخدام الحادثة MouseDown للتحقق من الضغط على أكثر من زر واحد.

أما بالنسبة للإجراء Form_MouseMove() فالأمر يختلف، إذ يمكن للوسيط Button أن يأخذ أي قيمة بين 0 و 7، مما يشمل كل الإمكانيات المحتملة لأزرار الفأرة الثلاث. فمثلاً، يساوي Button إلى 3 عند الضغط على الزرين الأيسر والأيمن والذي يكافئ الرقم الثنائي (00000011)، انظر الشكل 4-22.



الشكل 4-22

الوسيط Button للإجراء Form_MouseMove().

انظر الجدول التالي إذا لم تكن ملماً بالنظام الثنائي:

التدوين العشري Decimal Notation	التدوين الثنائي Binary Notation
0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
6	00000110
7	00000111

فمثلاً عندما تكون قيمة Button تساوي صفر، فالقيمة الثنائية تساوي (00000000) وحسب الشكل 4-22، الضغط على كل أزرار الفأرة سوياً تكون القيمة الثنائية المرافقة هي (00000111)، وعندما تكون القيمة مساوية إلى 4 فهذا معناه الضغط على الزر الأوسط والقيمة الثنائية المرافقة (00000100).

برنامج أزرار الفأرة 2

يوضح برنامج أزرار الفأرة 2 كيفية استخدام الوسيط Button، في الإجراء Form_MouseMove().

التمثيل المرئي لبرنامج أزرار الفأرة 2

سنبدأ بطور التمثيل المرئي لبرنامج أزرار الفأرة 2:

- أنشئ مشروعاً جديداً من نوع Standard EXE.
- احفظ نموذج المشروع بالاسم Button2.Frm في الدليل C:\VB5Prg\Ch04، واحفظ ملف المشروع بالاسم Button.Vbp في نفس الدليل.
- أنشئ نموذج البرنامج frmButton2 طبقاً للجدول 4-6.

الشكل 4 -23

النموذج frmButton2

في مرحلة التصميم.

الجدول 4-6. جدول خصائص النموذج frmButton2.

الكائن	الخاصية	القيمة
Form	**Name**	**frmButton2**
	Caption	برنامج أزرار الفأرة 2
	BackColor	White
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
Check Box	**Name**	**chkLeft**
	BackColor	White
	Caption	يساري
	Enabled	False
Check Box	**Name**	**chkMiddle**
	BackColor	White
	Caption	وسطي
	Enabled	False
Check Box	**Name**	**chkRight**
	BackColor	White
	Caption	يميني
	Enabled	False

Image	**Name**	**imgMouse**
	Picture	Mouse04.Ico
	Stretch	False
Label	**Name**	**lblInstruction**
	Caption	اضغط أحد أزرار الفأرة ثم حركها
	Alignment	2-Center
	BackColor	White
	BorderStyle	1-Fixed Single
	RightToLeft	True

▪يطالبك الجدول 4−6 بإسناد اسم الملف Mouse04.Ico إلى الخاصية Picture لعنصر التحكم Image.فإذا لم يكن هذا الملف موجوداً أو لم يكن الدليل Icons\Computer\ موجوداً لسبب ما، ضمن دليل لغة فيجول بيسك، تستطيع استخدام أي ملف آخر من نوع Ico.

▪يطالبك الجدول بإسناد القيمة False إلى الخاصية Stretch لعنصر التحكم Image. هذا يعني أن عنصر التحكم Image لن يوسِّع الصورة. فمثلاً إذا كانت الخاصية Picture لعنصر التحكم Image تشير إلى صورة بحجم Pixel 32x32، وكبرت بعدها حجم العنصر Image، فإن الصورة لن توسع ولن تغطي حجم عنصر التحكم Image. بل ستظهر الصورة بحجمها الأصلي فقط.

▪يسند الجدول القيمة False إلى الخاصية Enabled لكل مربع من مربعات الاختيار Check Boxes. مما يعني أن المستخدم لن يتمكن من اختيار هذه المربعات أو إلغاء اختيارها (أي حجبها) بالنقر بالفأرة عليها. وبدلاً من ذلك فإن نص البرنامج هو الذي سيختار أو يلغي اختيار مربعات التحكم بناءً على حالة أزرار الفأرة (وهو السبب وراء منع المستخدم من اختيار أو إلغاء هذه المربعات).

إدخال نص برنامج أزرار الفأرة 2

□تحقق من وجود العبارة التالية ضمن قسم التصاريح العامة General

:Declaration

```
' يجب التصريح عن كل المتحولات
Option Explicit
```

□أدخل النص التالي ضمن الإجراء ()Form_MouseMove:

```
Private Sub Form_MouseMove(Button As Integer, _
                 Shift As Integer, X As Single, Y As Single)
   If (Button And 1) = 1 Then
     chkLeft.Value = 1
   Else
     chkLeft.Value = 0
   End If

   If (Button And 2) = 2 Then
     chkRight.Value = 1
   Else
     chkRight.Value = 0
   End If

   If (Button And 4) = 4 Then
     chkMiddle.Value = 1
   Else
     chkMiddle.Value = 0
   End If
End Sub
```

□أدخل النص التالي ضمن الإجراء ()Form_MouseUp:

```
Private Sub Form_MouseUp(Button As Integer, _
                 Shift As Integer, X As Single, Y As Single)
   If Button = 1 Then
     chkLeft.Value = 0
   End If
   If Button = 2 Then
     chkRight.Value = 0
   End If
   If Button = 4 Then
     chkMiddle.Value = 0
   End If
End Sub
```

□أدخل النص التالي ضمن الإجراء ()cmdExit_ Click:

```
Private Sub cmdExit_Click()
    End
End Sub
```

□احفظ المشروع باختيار **Save Project** من القائمة **File** لفيجول بيسك.

تنفيذ برنامج أزرار الفأرة 2

□نفّذ برنامج أزرار الفأرة 2.

يبين الشكل 4-24 كيف يبدو إطار هذا البرنامج، لاحظ وجود ثلاثة مربعات اختيار: **يميني** و **وسطي** و **يساري**.



الشكل 4-24

إطار برنامج أزرار الفأرة2.

يختار برنامج أزرار الفأرة 2 ويلغي مربعات الاختيار تبعاً لحالة أزرار الفأرة " أثناء تحريك الفأرة ". فمثلاً يختار البرنامج مربعي الاختيار يميني و يساري إذا ضغط المستخدم على زري الفأرة اليساري واليميني وحرك الفأرة، (انظر الشكل 4-25).

كيف يعمل برنامج أزرار الفأرة 2

يستخدم برنامج أزرار الفأرة 2 الوسيط Button2 للإجراء ()Form_MouseMove لتحديد الزر أو مجموعة أزرار الفأرة التي ضغط عليها.



الشكل 4-25
برنامج أزرار الفأرة 2 عند الضغط على زري الفأرة اليساري واليميني (أثناء تحريكها).

نص الإجراء Form_MouseMove()

ينفذ الإجراء تلقائياً عند تحريك الفأرة ضمن منطقة نموذج البرنامج:

```
Private Sub Form_MouseMove(Button As Integer, _
                 Shift As Integer, X As Single, Y As Single)
    If (Button And 1) = 1 Then
      chkLeft.Value = 1
    Else
      chkLeft.Value = 0
    End If

    If (Button And 2) = 2 Then
      chkRight.Value = 1
    Else
      chkRight.Value = 0
    End If

    If (Button And 4) = 4 Then
      chkMiddle.Value = 1
    Else
      chkMiddle.Value = 0
    End If
End Sub
```

يختار نص هذا الإجراء أو يلغي مربعات الاختيار تبعاً لقيمة الوسيط Button. تحدد أول عبارة If في الإجراء إذا كان زر الفأرة الأيسر مضغوطاً وذلك بإجراء ضرب منطقي AND للوسيط Button مع القيمة 1.

فإذا كان ناتج حاصل الضرب يساوي 1 فهذا يعني أنه تم الضغط على الزر الأيسر، وبشكل مشابه إذا كان ناتج حاصل الضرب Button مع 2 منطقياً AND يساوي 2، فهذا معناه أنه تم الضغط على الزر الأيمن، وإذا كان ناتج الحاصل الضرب المنطقي AND لـــ Button مع 4 يساوي 4 فهذا دلالة على أن الزر الأوسط مضغوط حالياً، لا تكترث لحاصل الضرب المنطقي AND إذا لم تكن على اطلاع به أو لست على علم بالنظام الثنائي، المهم أن تعرف ما يحدث عند الضغط على أزرار الفأرة أثناء تحريكها.

نص الإجراء Form_MouseUp()

ينفذ هذا الإجراء عند تحرير أي من أزرار الفأرة:

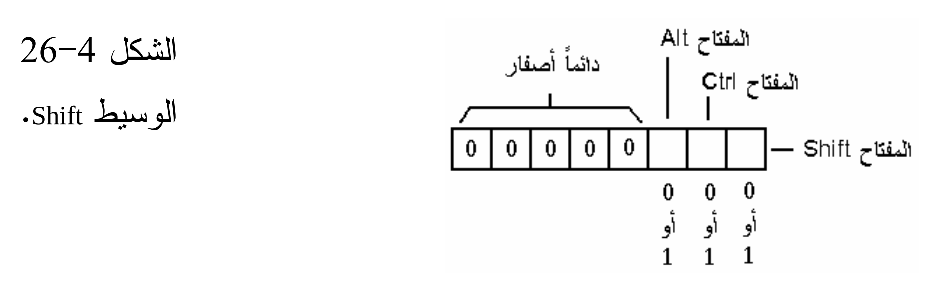
```
Private Sub Form_MouseUp(Button As Integer, _
               Shift As Integer, X As Single, Y As Single)
   If Button = 1 Then
     chkLeft.Value = 0
   End If
   If Button = 2 Then
     chkRight.Value = 0
   End If
   If Button = 4 Then
     chkMiddle.Value = 0
   End If
End Sub
```

يتحقق هذا الإجراء من الزر الذي حرر، ويلغي مربع الاختيار المرافق. فمثلاً تتحقق عبارة الشرط If الأولى عند تحرير الزر الأيسر للفأرة ويلغى اختيار مربع الاختيار **يساري**.

ضغط المفاتيح Shift و Ctrl و Alt إلى جانب ضغط أزرار الفأرة

تمتلك الحوادث MouseDown و MouseUp و MouseMove القيمة الصحيحة Shift كثاني وسيط ضمن الإجراءات المرافقة لها، ويشير هذا الوسيط إلى المفتاح Shift و Ctrl و Alt المضغوط إلى جانب أزرار الفأرة.

يبين الشكل 4-26 كيف يشير الوسيط Shift إلى المفتاح المضغوط.



الشكل 4–26
الوسيط Shift.

حسب ما يوضحه الشكل 4–26. تمثل البتات bits الأدنى من الوسيط Shift حالة المفاتيح Shift و Ctrl و Alt أثناء الضغط على زر (أو أزرار) الفأرة.

يوضح الجدول 4–7 القيم الثمان المحتملة التي يمكن أن يمتلكها الوسيط Shift.

الجدول 4–7 القيم المحتملة للوسيط Shift.

القيمة الثنائية	القيمة العشرية	المفتاح Shift	المفتاح Ctrl	المفتاح Alt
00000000	0	لا	لا	لا
00000001	1	نعم	لا	لا
00000010	2	لا	نعم	لا
00000011	3	نعم	نعم	لا
00000100	4	لا	لا	نعم
00000101	5	نعم	لا	نعم
00000110	6	لا	نعم	نعم
00000111	7	نعم	نعم	نعم

فمثلاً، إذا كانت قيمة الوسيط Shift تساوي 6 في أحد الإجراءات Form_MouseDown أو Form_MouseUp و Form_MouseMove فهذا يعني ضغط المفتاحين Ctrl و Alt إلى جانب ضغط أحد أزرار الفأرة.

لا تنس أن الوسيط Button يمكن أن يأخذ القيم 1 و 2 و 4 فقط في الإجراء Form_MouseDown(). بعكس الوسيط Shift لنفس الإجراء والذي يمكن أن يأخذ أي قيمة ضمن الجدول 4–7.

برنامج السحب

تعلمنا حتى الآن كيفية استخدام حوادث الفأرة التي تقع عند ضغط زر الفأرة MouseDown، أو عند تحرير زر فأرة MouseUp. أو عند تحريك الفأرة MouseMove. سنتعلم الآن كيف تستخدم حوادث الفأرة لبناء آلية تسمح للمستخدم بسحب وإسقاط (إفلات) عناصر التحكم ضمن إطار البرنامج.

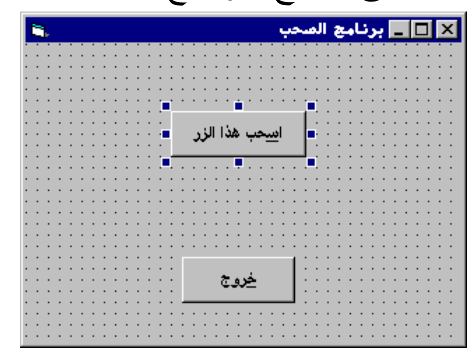
تعرف عملية السحب بأنها ضغط على الزر الأيسر للفأرة عندما يكون مؤشرها فوق عنصر تحكم، ثم تحريك الفأرة مع إبقاء الضغط على الزر. أما تحرير زر الفأرة بعد سحب العنصر فيدعى بالإفلات.

سنكتب الآن برنامج السحب، يوضح هذا البرنامج مدى سهولة إنجاز آلية السحب في البرنامج.

التمثيل المرئي لبرنامج السحب

سنشرع الآن بطور التمثيل المرئي لبرنامج السحب:

- أنشئ مشروعاً جديداً من نوع Standard EXE.
- احفظ نموذج المشروع بالاسم Drag.Frm في الدليل C:\VB5Prg\Ch04 واحفظ ملف المشروع بالاسم Drag.Vbp في نفس الدليل.
- أنشئ نموذج البرنامج طبقاً للجدول 4-8.



الشكل 4 -27
النموذج frmDrag
(طور التصميم).

الجدول 4-8. جدول خصائص برنامج السحب.

الكائن	الخاصية	القيمة
Form	**Name**	**frmDrag**

	Caption	برنامج السحب
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdDragMe**
	Caption	ا&سحب هذا الزر
	DragMode	1-Automatic
	RightToLeft	True

نص برنامج السحب

□تحقق من وجود العبارة Option Explicit ضمن قسم التصاريح العامة General Declaration للنموذج frmDrag:

```
' يجب التصريح عن كل المتحولات
Option Explicit
```

□أدخل النص التالي ضمن الإجراء cmdExit_ Click():

```
Private Sub cmdExit_Click()
    End
End Sub
```

□احفظ المشروع باختيار **Save Project** من القائمة **File** لفيجول بيسك.

تنفيذ برنامج السحب

تسند القيمة 1-Automatic إلى الخاصية DragMode للزر **اسحب هذا الزر**. يسمح هذا للمستخدم بسحب الزر أثناء زمن التنفيذ.

اتبع الخطوات التالية لرؤية كيفية سحب الزر **اسحب هذا الزر**:

□نفّذ برنامج السحب.

□اضغط زر الفأرة الأيسر عندما يكون مؤشر الفأرة فوق الزر **اسحب هذا الزر** واستمر بالضغط على الزر مع تحريك الفأرة.

كما تشاهد، يظهر مستطيل بحجم الزر ***اسحب هذا الزر*** *ويتبع حركة الفأرة.*

▫جرب سحب الزر **اسحب هذا الزر** خارج النموذج.

يستجيب البرنامج بإظهار دائرة مع خط مائل داخلها (رمز عملية غير نظامية)، وهي دلالة بأنك تحاول سحب عنصر التحكم إلى منطقة ممنوعة.

يختفي المستطيل عند تحرير زر الفأرة (تدعى هذه العملية بالإفلات). لاحظ أن الزر **اسحب هذا الزر** يبقى في موقعه الأصلي.

▫انقر على الزر **خروج** لبرنامج السحب لإنهائه.

تحسين برنامج السحب

كما لاحظنا، بسبب إسناد 1-Automatic إلى الخاصية DragMode يظهر مستطيل بنفس حجم الزر **اسحب هذا الزر**، ويتحرك استجابة لحركة الفأرة. يمكنك هذا المستطيل من مشاهدة المكان الذي يجري سحب الزر إليه.

لنولد شكلاً مختلفاً عند سحب عنصر التحكم، باتباع الخطوات التالية:

▫أسند اسم الملف Drag1Pg.Ico إلى الخاصية DragIcon للزر **اسحب هذا الزر** (يفترض أن تجد الملف Drag1Pg.Ico في الدليل \Icons\DragDrop تحت دليل لغة فيجول بيسك، ويمكنك استخدام أي ملف آخر من نوع Ico.* إذا لم تعثر على هذا الملف).

▫احفظ المشروع باختيار **Save Project** من القائمة **File**.

اتبع الخطوات التالية لرؤية تأثير تعديل الخاصية DragIcon:

▫ نفّذ برنامج السحب.

▫ اسحب الزر **اسحب هذا الزر**.

يستجيب البرنامج بإظهار الرمز Drag.Ico عند سحب الزر ***اسحب هذا الزر*** *(انظر الشكل 4 -28).*

والآن ستجد أن الرمز Drag1Pg.Ico سيتحرك تبعاً لحركة الفأرة بدلاً من المستطيل، قبل إجراء التعديل.



الشكل 4-28

الرمز Drag1Pg.Ico وهو يتحرك تبعاً لحركة الفأرة أثناء سحب الزر **اسحب هذا الزر**.

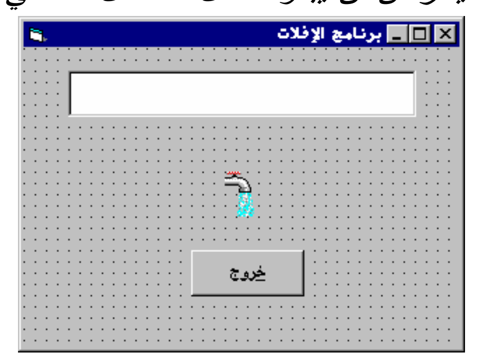
برنامج الإفلات

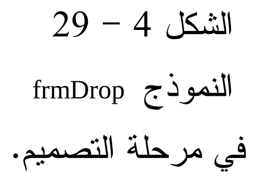
يوضح برنامج الإفلات مفهوم الإفلات (ويقصد به تحرير زر الفأرة بعد السحب) وكيف يستخدم في البرنامج.

التمثيل المرئي لبرنامج الإفلات

- أنشئ مشروعاً جديداً من نوع Standard EXE.
- احفظ نموذج المشروع بالاسم Drop.Frm في الدليل C:\VB5Prg\Ch04 واحفظ ملف المشروع بالاسم Drop.Vbp في نفس الدليل.
- أنشئ النموذج طبقاً للجدول 4-9.

يفترض أن يبدو الشكل المكتمل كما في الشكل 4-29.




الشكل 4 - 29

النموذج frmDrop في مرحلة التصميم.

الجدول 4-9. جدول خصائص برنامج الإفلات.

الكائن	الخاصية	القيمة
Form	**Name**	**frmDrop**
	Caption	برنامج الإسقاط

	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	خروج&
	RightToLeft	True
Text Box	**Name**	**txtInfo**
	Text	(اجعله فارغاً)
	Alignment	2-Center
	MultiLine	True
	Enabled	False
	RightToLeft	True
Image	**Name**	**imgWater**
	Picture	Water.Ico
	DragMode	1-Automatic
	Stretch	1-True
	Tag	Water image

يطالبك الجدول 4-9 بإسناد اسم الملف Water.Ico إلى الخاصية Picture لعنصر التحكم imgWater. تستطيع العثور على هذا الملف ضمن الدليل \Icons\Elements تحت دليل فيجول بيسك. استخدم أي ملف آخر من نوع Ico إذا لم تجد هذا الملف.

إدخال نص برنامج الإفلات

- تحقق من وجود العبارة Option Explicit ضمن قسم التصاريح العامة General Declaration:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء ()Form_DragOver:

```
Private Sub Form_DragOver(Source As Control, _
                  X As Single, Y As Single, State As Integer)
    Dim sInfo As String
    sInfo = "الآن يتم سحب "
    sInfo = sInfo + Source.Tag
```

```
    sInfo = sInfo + " فـوق الـنموذج "
    sInfo = sInfo + Str(State)
    txtInfo.Text = sInfo
End Sub
```

□أدخل النص التالي ضمن الإجراء cmdExit_ Click():

```
Private Sub cmdExit_Click()
    End
End Sub
```

□أدخل النص التالي ضمن الإجراء Form_DragDrop():

```
Private Sub Form_DragDrop(Source As Control, _
                             X As Single, Y As Single)
    'clear the text box.
    txtInfo.Text = ""
    'move the control
    Source.Move X, Y
End Sub
```

□أدخل النص التالي ضمن الإجراء cmdExit_DragOver():

```
Private Sub cmdExit_DragOver(Source As Control, _
                  X As Single, Y As Single, State As Integer)
    Dim sInfo
    sInfo = "الآن يـتم سحب "
    sInfo = sInfo + Source.Tag
    sInfo = sInfo + " فـوق الـزر خروج "
    sInfo = sInfo + Str(State)
    txtInfo.Text = sInfo
End Sub
```

□احفظ المشروع باختيار **Save Project** من القائمة **File** لفيجول بيسك.

تنفيذ برنامج الإفلات

□نفّذ برنامج الإفلات.

□اسحب صورة صنبور المياه.

تظهر رسالة ضمن مربع النص أثناء سحب الصورة، تشير إلى حالة عملية السحب.

يتسبب تحرير الفأرة بنقل الصورة إلى النقطة التي تم تحرير الفأرة فيها (نقطة

الإفلات).

- أنهِ البرنامج بالنقر على الزر **خروج**.

كيف يعمل برنامج الإفلات

يستخدم برنامج الإفلات الإجراءات Form_DragOver() و cmdExit_ DragOver() و Form_DragDrop().

نص الإجراء Form_DragOver()

ينفذ الإجراء Form_DragOver() عند سحب عنصر التحكم imgWater فوق النموذج. يمتلك هذا الإجراء أربعة وسائط Source و X و Y و Start:

```
Private Sub Form_DragOver(Source As Control, _
               X As Single, Y As Single, State As Integer)
   Dim sInfo As String
   sInfo = "الآن يتم سحب "
   sInfo = sInfo + Source.Tag
   sInfo = sInfo + " فوق النموذج "
   sInfo = sInfo + Str(State)
   txtInfo.Text = sInfo
End Sub
```

يمثل الوسيط Source اسم عنصر التحكم الذي يجري سحبه. وبما أن عنصر التحكم الذي يتم سحبه يدعى imgWater، وبالتالي يسند imgWater آلياً إلى الوسيط Source.

يمثل الوسيطان x و y الإحداثيات الراهنة x,y لمؤشر الفأرة (منسوبة إلى نظام إحداثيات النموذج). يمتلك الوسيط State إحدى القيم 0 أو 1 أو 2:

- تسحب الصورة من نقطة حرة إلى نقطة حرة أخرى ضمن النموذج، عندما تكون قيمة State تساوي 2.
- تسحب الصورة من نقطة حرة ضمن النموذج إلى نقطة غير نظامية (مثلاً نقطة خارج النموذج)، عندما تكون قيمة State تساوي 1.
- تسحب الصورة من نقطة غير نظامية إلى نقطة حرة ضمن النموذج، عندما

تساوي قيمة State الصفر.

يجهز الإجراء ()Form_DragOver سلسلة تدعى sInfo ويظهر السلسلة في مربع النص txtInfo:

```
Dim sInfo As String
sInfo = "الآن يتم سحب "
sInfo = sInfo + Source.Tag
sInfo = sInfo + " فوق النموذج "
sInfo = sInfo + Str(State)
txtInfo.Text = sInfo
```

فمثلاً يُسند إلى السلسلة ما يلي عند سحب الصورة من منطقة حرة ضمن النموذج إلى منطقة أخرى:

```
الآن يتم سحب  Water Image فوق النموذج 2
```

استخدمنا العبارة التالية أثناء تجهيز السلسلة sInfo:

```
sInfo = sInfo + Source.Tag
```

كما أسندنا أثناء طور التصميم، إلى الخاصية Tag لعنصر الصورة imgWater، القيمة Water Image، لذلك فإن قيمة Source.Tag ستساوي Water Image.

الخاصية Tag

تستخدم الخاصية Tag غالباً كمنطقة تخزين للمعطيات. فمثلاً، تحوي الخاصية Tag للعنصر imgWater على سلسلة مهمتها التعريف بعنصر التحكم هذا. استخدمنا في الإجراء ()Form_DragOver هذه السلسلة لتعريف العنصر المسحوب، طبعاً يمكن إسناد أي شيء آخر ترغب به (مثل MyWater) إلى الخاصية Tag.

نص الإجراء ()cmdExit_DragOver

ينفذ هذا الإجراء عند سحب عنصر التحكم فوق الزر **خروج**، ولهذا يتشابه هذا الإجراء مع الإجراء ()Form_DragOver باستثناء أن السلسلة sInfo تفيد بأن عنصر التحكم water يجري سحبه فوق الزر **خروج**.

```
Private Sub cmdExit_DragOver(Source As Control, _
                X As Single, Y As Single, State As Integer)
```

```
Dim sInfo As String
sInfo = "الآن يتم سحب "
sInfo = sInfo + Source.Tag
sInfo = sInfo + " فوق الزر خروج "
sInfo = sInfo + Str(State)
txtInfo.Text = sInfo
End Sub
```

فمثلاً ستحوي السلسلة الرسالة التالية عند سحب عنصر التحكم Water فوق الزر **خروج**:

```
الآن يتم سحب Water Image فوق الزر خروج 0
```

قيمة State = 0 هنا، وذلك بسبب سحب العنصر imgWater من خارج الزر **خروج** إلى داخله. ومن وجهة نظر الزر **خروج** فإن أي نقطة خارجه هي نقطة غير نظامية.

نص الإجراء ()Form_DragDrop

ينفذ هذا الإجراء عند إفلات عنصر التحكم ضمن النموذج. يجب إنجاز شيئين عند حدوث هذا: محو مربع النص، ونقل عنصر التحكم Water إلى نقطة الإفلات:

```
Private Sub Form_DragDrop(Source As Control, _
                            X As Single, Y As Single)
    'clear the text box.
    txtInfo.Text = ""
    'move the control
    Source.Move X, Y
End Sub
```

لهذا استخدمنا العبارتين التاليتين ضمن الإجراء:

```
txtInfo.Text = ""
Source.Move X, Y
```

ينقل عنصر التحكم imgWater إلى نقطة الإفلات باستخدام الطريقة Move. تتحدد نقطة الإفلات بالوسيطين X و Y لهذا الإجراء.

الخلاصة

تناول هذا الفصل حوادث الفأرة. فتعلمنا الحوادث MouseDown و MousMove

وDragOver و DragDrop.

تقدم وسائط إجراءات هذه الحوادث معلومات عن حالة الفأرة زمن وقوع الحادثة. فالوسيط Button مثلاً يشير إلى زر الفأرة الذي تم النقر عليه والوسيط Shift يخبر أي مفتاح من المفاتيح Shift أو Ctrl أو Alt تم الضغط عليه إلى جانب الفأرة. كما يحدد الوسيطان X و Y موقع مؤشر الفأرة زمن وقوع الحادثة، وهكذا.

الفصل الخامس

إنشاء القوائم

يتركز اهتمام هذا الفصل، على آلية دمج القوائم في برامجك. وسوف تتعرف من خلاله على الطريقة التي يتم بها تصميم القائمة وكيفية ربطها مع البرنامج.

برنامج الألوان

سنبدأ الآن بكتابة برنامج يتضمن قائمة Menu، وسندعوه برنامج الألوان. يسمح لك برنامج الألوان باختيار لون من قائمة، وتلوين خلفية البرنامج باللون المنتقى. كما يسمح باختيار حجم إطار البرنامج من قائمة أخرى.

يلزمنا قبل البدء بكتابة برنامج الألوان، تحديد كيف ستبدو القائمة في البرنامج وما الذي يفترض أن تفعله:



الشكل 5-1

برنامج الألوان

- سيظهر لك عند تشغيل برنامج الألوان، شريط قوائم. يحمل عنواني قائمتين هما: قائمة **الألوان** وقائمة **الحجم**. (انظر الشكل 5-1).
- تحتوي قائمة **الألوان** على بندين هما:بند **تحديد اللون** وبند **خروج** (انظر الشكل 5-2).



الشكل 5-2

بندي قائمة الألوان

□كما تحوي قائمة **الحجم** على بندين أيضاً، هما: البند **صغير** والبند **كبير**.



الشكل 5-3

قائمة الحجم.

□تظهر قائمة أخرى لدى اختيار البند **تحديد اللون** من قائمة **الألوان**. تُظهر هذه القائمة لائحةً بالألوان التي يمكنك الانتقاء من بينها. يتغير لون خلفية نموذج البرنامج إلى اللون الجديد فور اختياره. (انظر الشكل 5-4).

□كذلك، يتغير حجم إطار برنامج الألوان، تبعاً للبند الذي تنتقيه من قائمة **الحجم**.

□يتسبب اختيار البند **خروج**، من قائمة **الألوان** بإنهاء البرنامج.

التمثيل المرئي لبرنامج الألوان

سنبدأ كعادتنا عند تصميم البرنامج، ببناء نموذج البرنامج:

□أنشئ الدليل الفرعي التالي: C:\VB5Prg\Ch05.

□ابدأ مشروعاً جديداً Project من النوع Standard EXE.

□احفظ نموذج المشروع باسم Colors.Frm في الدليل الفرعي C:\VB5Prg\Ch05، ثم احفظ ملف المشروع باسم Colors.Vbp في ذات الدليل.

□أنشئ نموذج برنامج الألوان طبقاً للجدول 5-1.

الجدول 5-1. جدول خصائص برنامج الألوان.

الكائن	الخاصية	القيمة
Form	Name	frmColors

	Caption	برنامج الألوان
	RightToLeft	True
	BackColor	White

بناء قوائم برنامج الألوان

سنعمل الآن على إنشاء قوائم برنامج الألوان. تحتاج القائمة لنموذج ترتبط معه، لهذا يجب علينا اختيار النموذج الذي سترتبط به القائمة، قبل بناء هذه القائمة:

- تأكد أن النموذج frmColors هو الإطار المنتقى. (قد تبدو هذه الخطوة لا لزوم لها في برنامجنا هذا، على اعتبار أن مشروعنا لا يحتوي سوى نموذجاً واحداً فقط). ولكن تغدو هذه الخطوة ضرورية عندما يحوي المشروع على أكثر من نموذج واحد، لأنه سيتوجب عليك ربط القائمة بالنموذج الذي تحدده.

ملاحظة

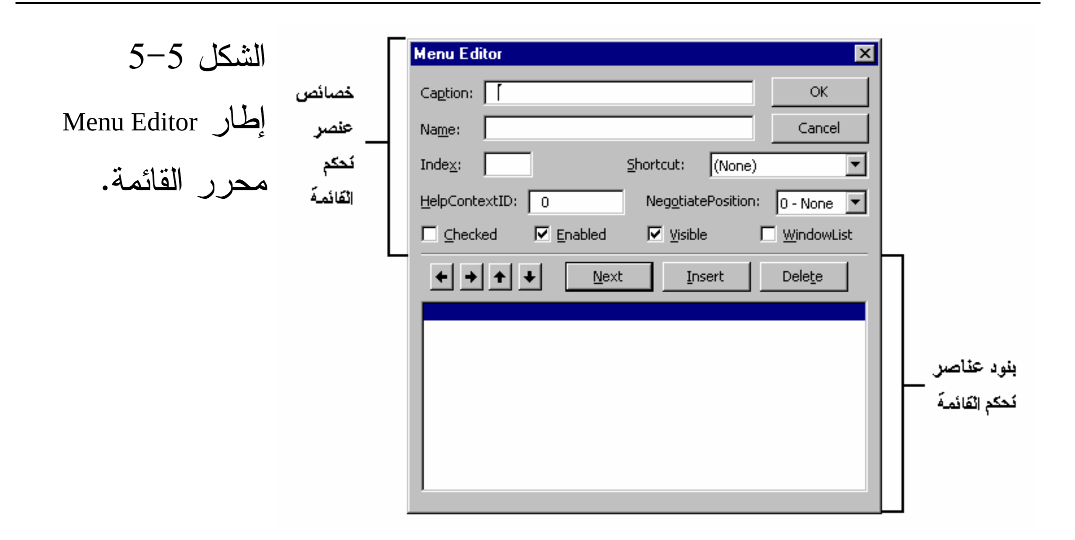
يؤدي النقر بالزر الأيسر للفأرة داخل النموذج أو أي عنصر آخر إلى اختيار هذا النموذج.

الآن، وبعد أن انتهينا من اختيار النموذج frmColors، سيربط فيجول بيسك القائمة بالنموذج frmColors.

- اختر البند Menu Editor من القائمة Tools.

يستجيب فيجول بيسك بإظهار إطار Menu Editor المبين في الشكل 5-5.

سنستخدم إطار Menu Editor من الآن فصاعداً، لبناء نظام القائمة المبين في الشكل5-1.



الشكل 5-5
إطار Menu Editor
محرر القائمة.

إنشاء عناصر تحكم القائمة

حسب ما يوضحه الشكل 5-5 فإن مربع حوار Menu Editor يقسم إلى جزئين: الجزء السفلي وهو لائحة بأسماء عناصر تحكم القائمة Menu Control List Box، والجزء العلوي ويعرض خصائص كل عنصر Menu Control Properties.

يحتوي الجزء السفلي عادةً، كل عناصر تحكم القائمة الموجودة في البرنامج، مثلاً، سيحتوي هذا الجزء في مثالنا الحالي على كل عناصر تحكم القائمة الموجودة في برنامج الألوان، السؤال هو: ما هي عناصر تحكم القائمة ؟.

يمكن أن تكون عناصر تحكم القائمة عبارة عن عناوين قوائم. (مثل عنوان قائمة الألوان وعنوان قائمة **الحجم**، المبينين في الشكل 5-1). أو قد تكون بنود قوائم (مثل البند **تحديد اللون** والبند **خروج**، المبينين في الشكل 5-2).

يمتلك برنامج الألوان كما رأينا قائمتين منسدلتين هما: قائمة **الألوان** المبينة في الشكل 5-2 وقائمة **الحجم** المبينة في الشكل 5-3.

تمتلك قائمة **الألوان** بدورها ثلاثة عناصر تحكم قائمة هي:

الألوان (عنوان القائمة).

تحديد اللون (بند قائمة).

خروج (بند قائمة).

حسب ما يوضحه الشكل 5-2.

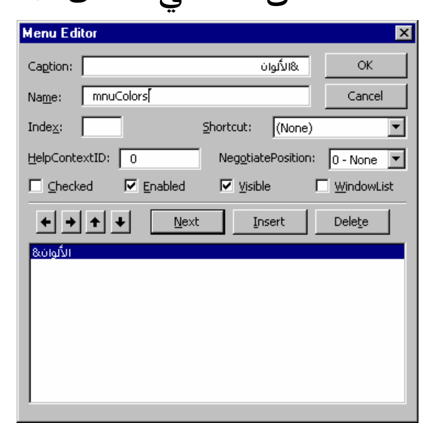
يظهر حتى هذه النقطة، شريط غامق في الجزء السفلي من **محرر القائمة**. كما هو واضح في الشكل 5-5. وهذا يعني أن فيجول بيسك صار جاهزاً لتشكيل عناصر تحكم القائمة.

اتبع الخطوات التالية لإنشاء عنوان قائمة **الألوان**:

- اكتب "&الألوان" في الحقل النصي Caption. بدون كتابة إشارات التنصيص.
- اكتب "mnuColors" في الحقل النصي Name. بدون كتابة إشارات التنصيص.

اتفقنا أن الرمز & قبل أحد الأحرف (للخاصية Caption فقط)، يعني أن الحرف الذي يأتي وراء هذا الرمز سيظهر وتحته خط. فمثلاً ستظهر قائمة **الألوان** مع خط قصير تحت الحـــــرف " ا "، هكذا " ا ". مما يعني أنه يمكن الضغط على المفتاحين "Alt+"ا"، عند تنفيذ البرنامج لاختيار قائمة **الألوان** بدلاً من النقر على عنوانها بواسطة الزر الأيسر للفأرة.

انتهينا للتو من إنشاء عنوان لقائمة الألوان. يفترض أن يظهر الـــ Menu Editor الآن كما في الشكل 5-6.



الشكل 5-6

إنشاء عنوان

قائمة الألوان

الخطوة التالية الآن إنشاء بندي قائمة **الألوان**.

اتبع الخطوات التالية لإنشاء البند تحديد اللون في قائمة **الألوان**:

- انقر الزر **Next** في إطار Menu Editor.

يستجيب فيجول بيسك بإضاءة السطر التالي في الجزء السفلي من إطار Menu Editor.

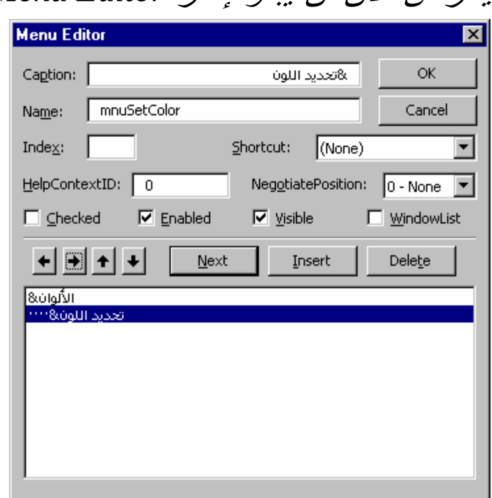
- اكتب "&تحديد اللون" في الحقل النصي Caption من الإطار Menu Editor.
- اكتب "mnuSetColor" في الحقل النصي Name من الإطار Menu Editor.

باعتبار أن بند **تحديد اللون** موجود في قائمة **الألوان**، إذاً لا بد من إزاحته وجعله كفرع ثانوي من الفرع الأصلي. اتبع الخطوات التالية لإزاحة بند **تحديد اللون**:

- انقر زر السهم اليميني في الإطار Menu Editor.

*استجابة لذلك، يزاح بند **تحديد اللون** إلى اليمين. (تظهر ثلاث نقاط على يساره).*

يفترض الآن أن يبدو إطار Menu Editor كما في الشكل 5-7.



الشكل 5-7

إنشاء بند تحديد اللون

في قائمة الألوان.

يضع فيجول بيسك خطاً قصيراً تحت الحرف "ت"، نتيجة لاستخدام الرمز & في كتابة قيمة الخاصية Caption لبند **تحديد اللون**. وعند تشغيل البرنامج يؤدي الضغط على حرف "ت" بعد إظهار قائمة **الألوان**، إلى نفس النتيجة التي تنشأ عن اختيار بند **تحديد اللون** عن طريق الفأرة.

استخدم الخطوات التالية لإنشاء البند الثاني في قائمة الألوان:

- انقر الزر **Next** في الإطار Menu Editor.

يستجيب فيجول بيسك بإضاءة السطر التالي.

- اكتب "&خروج" في الحقل النصي Caption من الإطار Menu Editor.

❑ اكتب "mnuExit" في الحقل النصي Name من الإطار Menu Editor.

يستجيب فيجول بيسك بإضافة هذا البند آلياً تحت سابقه، بمعنى أنك لست بحاجة إلى إزاحته كما فعلت مع البند الأول.

هذا كل شيء بالنسبة لقائمة **الألوان**.

والآن جاء دور إنشاء قائمة **الحجم**. تتضمن هذه القائمة حسب ما أوضحه الشكل 5-3، ثلاثة عناصر تحكم قائمة هي:

الحجم (عنوان القائمة).

صغير (بند قائمة).

كبير (بند قائمة).

استخدم الخطوات التالية لإنشاء عنوان قائمة **الحجم**:

❑ انقر الزر **Next** في الإطار Menu Editor.

*يستجيب فيجول بيسك بإضاءة السطر التالي لسطر آخر بند تم إنشاؤه (بند **خروج**).*

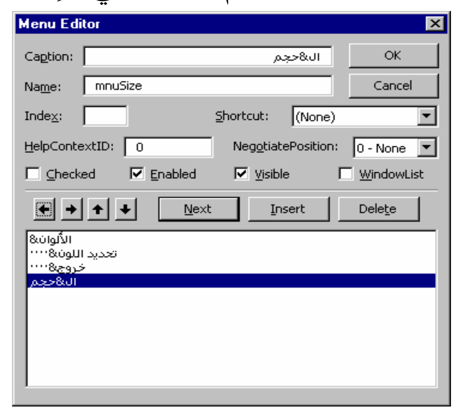
❑ اكتب "ال&حجم" في الحقل النصي Caption.

❑ اكتب "mnuSize" في الحقل النصي Name.

يزيح فيجول بيسك هذا البند تلقائياً، ولكن بما أن هذا البند ليس بنداً عادياً، بل هو عنوان قائمة، فإنه لا بد من التراجع عن هذه الإزاحة (نقر زر السهم اليساري).

اتبع هذه الخطوة لإزالة الإزاحة الموجودة قبل عنوان قائمة **الحجم**:

❑ انقر زر السهم الأيسر في الإطار Menu Editor، مثلما هو مبين في الشكل5-8.



الشكل 5-8

إنشاء عنوان

قائمة الحجم.

اتبع الخطوات التالية لإنشاء بندي قائمة **الحجم:**

- انقر الزر **Next** في الإطار Menu Editor.

يستجيب فيجول بيسك بإضاءة السطر التالي لسطر آخر بند تم إنشاؤه(عنوان قائمة الحجم).

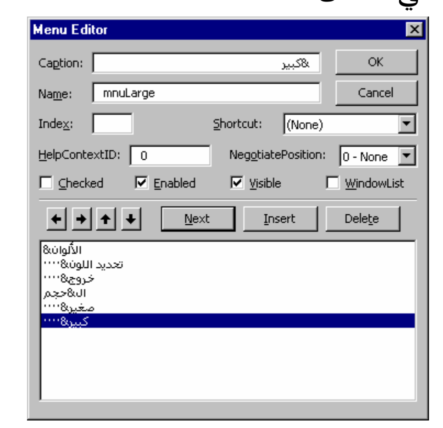
- اكتب "&صغير" في مربع النص Caption.
- اكتب "mnuSize" في الحقل النصي Name.

بما أن البند **صغير** هو بند في قائمة **الحجم**، فلا بد من إزاحته، وجعله في المستوى الثاني. اتبع ما يلي:

- انقر زر السهم اليميني في الإطار Menu Editor. (تظهر ثلاث نقاط على يسار البند **صغير**).
- استخدم الآن الخطوات التالية لإنشاء البند الثاني في قائمة **الحجم:**
- انقر الزر **Next** في الإطار Menu Editor.
- اكتب "&كبير" في مربع النص Caption.
- اكتب "mnuSize" في مربع النص Name.

لا داعي لإزاحة هذا البند، لأن فيجول بيسك يعتبره بنفس مستوى البند السابق له، وهو المطلوب.

أصبحت قائمة الحجم جاهزة هي الأخرى! ويفترض أن يظهر الإطار Menu Editor كما في الشكل 5-9.



الشكل 5-9

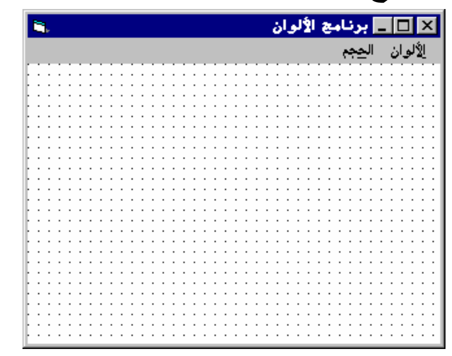
إنشاء بنود

قائمة الحجم.

لم تنته مرحلة تصميم القائمة بعد، ولكن دعنا نشاهد ما أنجزناه حتى هذه اللحظة!

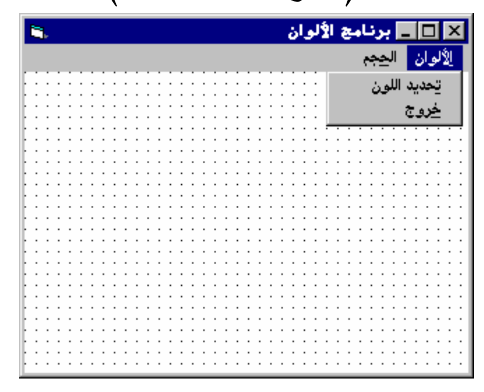
- انقر الزر **Ok** في الإطار Menu Editor.
- احفظ العمل المنجز باختيار البند **Save Project** من قائمة **File**.

يوضح الشكل 5-10 كيف يظهر النموذج frmColors حتى الآن.



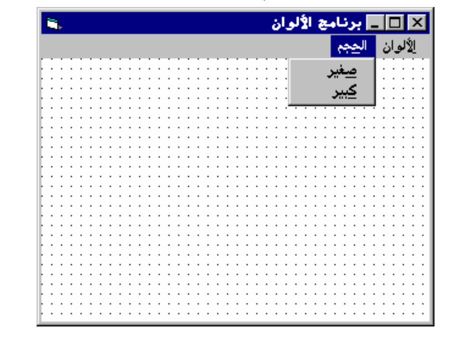
الشكل 5-10
النموذج frmColors
مع شريط قائمة
(مرحلة التصميم).

- انقر على عنوان قائمة **الألوان** فوق شريط القوائم، تنسدل عندها لائحة ببنود هذه القائمة (انظر الشكل 5-11).



الشكل 5-11
بنود قائمة الألوان
(مرحلة التصميم).

- انقر على عنوان قائمة **الحجم** فوق شريط القوائم، لفتح هذه القائمة(انظر الشكل5-12).



الشكل 5-12
قائمة الحجم
(مرحلة التصميم).

دعنا ننفذ برنامج الألوان:

- نفّذ برنامج الألوان. (اضغط المفتاح F5).

*يظهر إطار برنامج الألوان. وكما تلاحظ يحتوى إطار البرنامج على شريط قوائم، يحمل عنواني قائمتين: قائمة **الألوان** وقائمة **الحجم**.*

- انقر عنوان قائمة **الألوان**.

*يستجيب برنامج الألوان بإظهار قائمة **الألوان**.*

- انقر عنوان قائمة **الحجم**.

*يستجيب برنامج الألوان بإظهار قائمة **الحجم**.*

تستطيع طبعاً نقر أي بند من بنود القائمتين لاختياره، ولكن لن يحدث أي شيء، والسبب طبعاً أننا لم نكتب حتى الآن أي نص برمجي، ولم نربط بعد أي نص برمجي مع بنود القائمتين.

- انقر على الرمز الذي يظهر في أقصى الزاوية اليمنى العليا من برنامج الألوان (إشارة x)، وذلك لإنهاء عمل البرنامج.

*يستجيب برنامج الألوان وينهي نفسه. (طبعاً، قمنا بهذا العمل لأننا لم نربط أي نص برنامج مع البند **خروج**، ولهذا لا فائدة حالياً من نقر بند **خروج** الموجود في قائمة الألوان لأنه لن يؤدي إلى إنهاء تنفيذ البرنامج).*

إنشاء قائمة ثانوية

سوف نستأنف الآن عملية التصميم المرئي لنظام القائمة في برنامج الألوان. لنبدأ الآن بقائمة **الألوان** (انظر الشكل 5−11).

تتضمن هذه القائمة بندين هما:

تحديد اللون.

خروج.

سنعمل لاحقاً، على ربط البند **خروج** بنص البرنامج المناسب، بحيث يؤدي النقر على هذا البند إلى قيام البرنامج بإنهاء نفسه. ما الذي سيحدث لدى قيام المستخدم بالنقر

على بند **تحديد اللون؟** الجواب هو ظهور قائمة أخرى (تدعى قائمة ثانوية أو فرعية Submenu). سنشرع فوراً ببنائها.

يجب أن تكون محتويات قائمة تحديد اللون الثانوية كالتالي:

أحمر.

أزرق.

أبيض.

□يجب طبعاً أن تكون نافذة النموذج frmColors منتقاة. اختر **Menu Editor** من القائمة **Tools** لفيجول بيسك.

يستجيب فيجول بيسك بإظهار الإطار Menu Editor.

البند **أحمر** هو أحد بنود القائمة الفرعية التابعة للبند **تحديد اللون**. ولهذا سنحتاج إلى حشره (إضافته) بين بند **تحديد اللون** والبند **خروج**.

□اختر البند **خروج** (دع الإضاءة تتوضع عنده). لأنك على وشك حشر بند إضافي في السطر الذي يقع فوق البند **خروج**.

□انقر الزر **Insert** في الإطار Menu Editor.

*يستجيب فيجول بيسك بحشر سطر غامق تحت البند **تحديد اللون**.*

□اكتب "أح&مر" في مربع النص Caption.

□ اكتب "mnuRed" في مربع النص Name.

كما تلاحظ، يقع البند **أحمر** الآن في نفس مستوى بند **تحديد اللون**، إلا أنه يجب إزاحته، لأنه في الواقع ينتمي إلى قائمة **تحديد اللون** الثانوية.

استخدم الخطوة التالية لإزاحة البند أحمر.

□ انقر زر السهم اليميني في الإطار Menu Editor.

استخدم الخطوات التالية لإضافة البند **أزرق:**

□ اختر البند **خروج**، ثم انقر الزر **Insert** في الإطار Menu Editor، لإضافة وإضاءة سطر غامق (خالي) تحت بند القائمة الثانوية المسمى أحمر.

□ اكتب "أزر&ق" في مربع النص Caption.

- اكتب "mnuBlue" في مربع النص Name.
- انقر زر السهم اليميني في الإطارMenu Editor لوضع البند **أزرق** تحت نفس مستوى إزاحة البند **أحمر**.

اتبع الخطوات التالية لإضافة البند أبيض:

- اختر البند خروج، ثم انقر الزر **Insert** في الإطار Menu Editor، لإضافة وإضاءة سطر غامق (خالي) تحت بند القائمة الثانوية المسمى أزرق.
- اكتب "أبي&ض" في حقل النص Caption.
- اكتب "mnuWhite" في حقل النص Name.
- انقر زر السهم اليميني في الإطار Menu Editor لإزاحة البند **أبيض** إلى نفس سوية البندين **أحمر** و **أزرق**.
- كرر نفس الخطوات لإضافة البندين **صغير** و**كبير** إلى قائمة **الحجم**.

تهانينا**!** لقد انتهينا من بناء قائمة برنامج الألوان! ويفترض أن يظهر الإطار Menu Editor كما في الشكل 5-13.

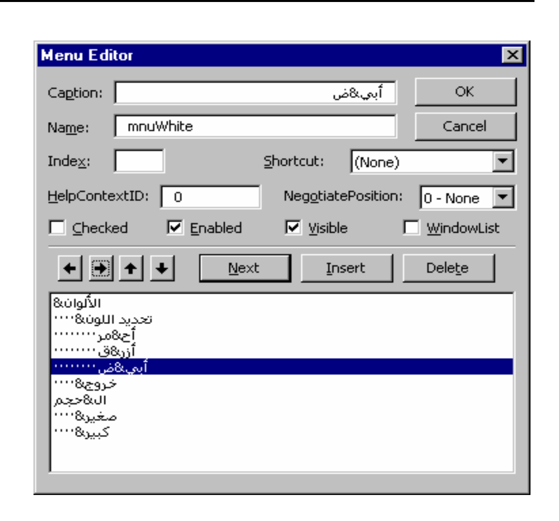
استخدم الخطوات التالية للخروج من Menu Editor:

- انقر الزر **Ok** في الإطار Menu Editor.

يستجيب فيجول بيسك بإغلاق Menu Editor، ويظهر النموذج frmColors. وكما تشاهد تم ربط القائمة التي انتهيت لتوك من تصميمها بالنموذج.

اكتمل الآن التصميم المرئي لبرنامج الألوان.

- احفظ المشروع باختيار البند **Save Project** من قائمة **File**.



الشكل 5-13

إطار Menu Editor

بعد اكتمال قوائم

برنامج الألوان.

إنشاء قائمة وفقاً لجدول معين

لقد شرحنا خطوة خطوة، كيفية بناء قوائم برنامج الألوان، لأنه أول برنامج لك في هذا الصدد. لا يمكننا طبعاً الاستمرار على هذا المنوال في الفصول المقبلة، وإنما سنلجأ من الآن فصاعداً إلى تزويدك بجدول يساعدك على إنشاء تلك القوائم.

يتألف جدول القائمة من عمودين، يدرج في العمود اليميني عناوين Captions ومستويات إزاحة البنود، بينما تدرج أسماء البنود (أسماء عناصر تحكم النافذة) في العمود اليساري.

يوضح الجدول 5-2 نموذجاً لجدول قائمة. وسنكتشف بسرعة، أن هذا الجدول هو جدول برنامج الألوان.

الجدول 5–2. جدول قائمة برنامج الألوان.

العنوان Caption	الخاصية Name
&الألوان	mnuColors
...&تحديد اللون	mnuSetColors
......أح&مر	mnuRed
......أزر&ق	mnuBlue
......أبي&ض	mnuWhite
...&خروج	mnuExit
ال&حجم	mnuSize
...&صغير	mnuSmall
...&كبير	mnuLarge

إدخال نص برنامج الألوان

يمتلك كل بند من بنود القائمة التي صممتها للتو، (بغض النظر عن مستواه) حادثة نقر Click Event. تنفذ هذه الحادثة لدى اختيار بند القائمة المرافق لها، فمثلاً تنفذ الحادثة mnuExit_Click() تلقائياً، عند اختيار البند **خروج** من قائمة **الألوان**. يبدأ اسم الإجراء لهذه الحادثة بالرموز mnuExit، التي تمثل الاسم الذي أطلقته على بند القائمة **خروج** في الإطار Menu Editor.

رغم أن الحادثة تدعى Click (أي نقر) إلا أنها تنفذ سواء تم اختيار البند بنقره بواسطة الفأرة، أو ضغط أحرف الوصول لهذا البند، بواسطة لوحة المفاتيح.

لإظهار الإجراء الخاص ببند قائمة ما، انقر على ذلك البند (في مرحلة التصميم). مثلاً، انقر على البند **خروج**، إذا أردت مشاهدة الإجراء الذي ينفذ عند اختيار هذا البند.

هناك طريقة أخرى لإظهار الإجراء الخاص ببند قائمة ما، وتكون بإظهار إطار نص البرنامج (Code Window) التابع للنموذج، بالنقر المزدوج على أي مكان في النموذج، ثم اختيار بند القائمة المطلوب، من مربعي السرد الموجودين في قمة إطار نص البرنامج.

اكتب الآن نص برنامج الألوان:

- تحقق أن قسم التصاريح العامة General Declarations للنموذج frmColors يحوي العبارة Option Explicit، أي:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

سيطلب منك في الخطوة التالية إدخال نص البرنامج داخل الإجراء Form_Load(). ولإظهار هذا الإجراء انقر نقراً مزدوجاً على النموذج، ثم ضع الإضاءة في مربع السرد اليساري على Form، وضع مربع السرد الأيمن على Load، فيُظهر فيجول بيسك حينها الإجراء Form_Load().

- أدخل نص البرنامج التالي في الإجراء Form_Load():

```
Private Sub Form_Load()
    'بما أن لون الإطار الابتدائي هو الأبيض
    'فيجب تعطيل البند "أبيض" في القائمة
    mnuWhite.Enabled = False

    'بما أن حجم الإطارالابتدائي هو صغير
    'فيجب تعطيل البند "صغير" في القائمة
    mnuSmall.Enabled = False
End Sub
```

ملاحظة

سيطلب منك في الخطوة التالية إدخال نص البرنامج للإجراء mnuRed_Click(). يمكنك إظهار هذا الإجراء، بإظهار النموذج frmColors، ثم اختيار بند **تحديد اللون** من قائمة **الألوان**، وأخيراً اختيار البند **أحمر** من القائمة الثانوية التي ستظهر.

أو تستطيع إظهار الإجراء ()mnuRed_Click باتباع الخطوات التالية:

- إظهار إطار نص البرنامج.
- وضع مربع السرد اليساري من مربع إطار نص البرنامج على mnuRed.
- وضع مربع السرد اليميني من إطار نص البرنامج على Click.

□أدخل نص البرنامج التالي في الإجراء ()mnuRed_Click:

```
Private Sub mnuRed_Click()
    'تغيير لون النموذج إلى الأحمر'
    frmColors.BackColor = vbRed

    'بما أن لون الإطار صار أحمر'
    'يجب تعطيل البند "أحمر" وتفعيل'
    'البندين "أبيض" و "أزرق" في القائمة'
    mnuRed.Enabled = False
    mnuBlue.Enabled = True
    mnuWhite.Enabled = True
End Sub
```

□أدخل نص البرنامج التالي في الإجراء ()mnuBlue_Click:

```
Private Sub mnuBlue_Click()
    'تغيير لون النموذج إلى الأزرق'
    frmColors.BackColor = vbBlue

    'بما أن لون الإطار صار أزرق'
    'يجب تعطيل البند "أزرق" وتفعيل'
    'البندين "أبيض" و "أحمر" في القائمة'
    mnuBlue.Enabled = False
    mnuRed.Enabled = True
    mnuWhite.Enabled = True
End Sub
```

□أدخل نص البرنامج التالي في الإجراء ()mnuWhite_Click:

```
Private Sub mnuWhite_Click()
    'تغيير لون النموذج إلى الأبيض'
    frmColors.BackColor = vbWhite

    'بما أن لون الإطار صار أبيض'
```

```
    'يجب تعطيل البند "أبيض" وتفعيل
    'البندين "أحمر" و "أزرق" في القائمة
    mnuWhite.Enabled = False
    mnuBlue.Enabled = True
    mnuRed.Enabled = True
End Sub
```

□أدخل نص البرنامج التالي في الإجراء ()mnuSmall_Click:

```
Private Sub mnuSmall_Click()
    'تغيير حجم النموذج إلى صغير
    frmColors.WindowState = vbNormal

    'تعطيل بند القائمة صغير
    mnuSmall.Enabled = False

    'تفعيل بند القائمة كبير
    mnuLarge.Enabled = True
End Sub
```

□أدخل نص البرنامج التالي في الإجراء ()mnuLarge_Click:

```
Private Sub mnuLarge_Click()
    'تغيير حجم النموذج إلى كبير
    frmColors.WindowState = vbMaximized

    'تعطيل بند القائمة كبير
    mnuLarge.Enabled = False

    'تفعيل بند القائمة صغير
    mnuSmall.Enabled = True
End Sub
```

□أدخل نص البرنامج التالي في الإجراء ()mnuExit_Click:

```
Private Sub mnuLarge_Click()
    End
End Sub
```

□اختر البند **Save Project** من القائمة **File** لحفظ العمل المنجز حتى الآن.

تنفيذ برنامج الألوان

نفّذ برنامج الألوان واعمل على اختيار مختلف بنود القائمة، ستلاحظ المزايا التالية أثناء تنفيذك للبرنامج:

يكون لون البند **أبيض** المنتمي إلى القائمة الثانوية **تحديد اللون**، بلون باهت (بمعنى أنه معطلاً أو ليس فعالاً) عند تنفيذ البرنامج. وهذا طبيعي، لأن لون النموذج الافتراضي هو اللون الأبيض عند بداية التشغيل. ولذلك لا فائدة من إجراء تغيير لون النموذج إلى اللون الأبيض طالما أن لونه بالأساس هو اللون الأبيض.

يقاس نفس الشيء بالنسبة للبند **صغير**، المنتمي إلى قائمة **الحجم**. وسيكون لونه عند بداية التشغيل باهتاً، لأن النموذج صغير بالأصل.

يغير النموذج لونه بعد اختيار أحد ألوان قائمة البند **تحديد اللون**. ويصبح لون البند الذي اخترته من قائمة **تحديد اللون** باهتاً (بلون رمادي باهت).

□اختر البند **خروج** من قائمة **الألوان**، لإنهاء عمل برنامج الألوان.

كيف يعمل برنامج الألوان

يستخدم برنامج الألوان الإجراء ()Form_Load، وأيضاً الحادثة Click لبنود القوائم المختلفة.

نص برنامج الإجراء ()Form_Load:

يعتبر الإجراء ()Form_Load من أول الإجراءات التي تنفذ آلياً عند بدء تشغيل أي برنامج مطوّر في فيجول بيسك. يمكنك استخدام الإجراء ()Form_Load لإنجاز مختلف مهام الابتداء التي ترغب أن ينجزها برنامجك عند بداية التنفيذ.

في مثالنا هذا، يفترض أن تكتب نص برنامج الإجراء ()Form_Load، بشكل يلغي عمل (يعطّل) البند **أبيض** المنتمي إلى القائمة الثانوية **تحديد اللون**. كما يلغي عمل البند **صغير** المنتمي إلى قائمة **الحجم**.

يعطل الإجراء Form_Load() عمل البندين **أبيض** و **صغير** بإسناد القيمة False للخاصية Enabled لكلا البندين.

```
Private Sub Form_Load()
    'بما أن لون الإطار الابتدائي هو الأبيض
    'فيجب تعطيل البند "أبيض" في القائمة
    mnuWhite.Enabled = False

    'بما أن حجم الإطار الابتدائي هو صغير
    'فيجب تعطيل البند "صغير" في القائمة
    mnuSmall.Enabled = False
End Sub
```

نص برنامج الإجراء mnuRed_Click()

ينفذ الإجراء mnuRed_Click() لدى اختيار البند **أحمر** من القائمة الثانوية للبند **تحديد اللون**. ويؤول لون النموذج إلى اللون الأحمر، ويتم تعطيل البند **أحمر** في قائمة البند **تحديد اللون**:

```
Private Sub mnuRed_Click()
    'تغيير لون النموذج إلى الأحمر
    frmColors.BackColor = vbRed

    'بما أن لون الإطار صار أحمر
    'يجب تعطيل البند "أحمر" وتفعيل
    'البندين "أبيض" و "أزرق" في القائمة
    mnuRed.Enabled = False
    mnuBlue.Enabled = True
    mnuWhite.Enabled = True
End Sub
```

يعمل الإجراء mnuRed_Click() على تعطيل البند **أحمر**. وتفعيل البندين **أزرق** و **أبيض**، وباعتبار أن لون النموذج صار أحمر، فمن المفترض أن يكون لدى المستخدم القدرة على تغيير لون النموذج إلى الأزرق أو الأبيض. يعتبر عمل الإجراءين mnuRed_Click() و mnuWhite_Click() مشابهاً جداً لعمل الإجراء mnuRed_Click().

نص برنامج الإجراء ()mnuSmall_Click

ينفذ الإجراء ()mnuRed_Click لدى اختيار البند **صغير** من قائمة **الحجم**، ويؤدي إلى تغيير حجم النموذج إلى مقاس صغير. ثم يعطل عمل البند **صغير**.

```
Private Sub mnuSmall_Click()
    'تغيير حجم النموذج إلى صغير
    frmColors.WindowState = vbNormal

    'تعطيل بند القائمة صغير
    mnuSmall.Enabled = False

    'تفعيل بند القائمة كبير
    mnuLarge.Enabled = True
End Sub
```

تُستخدم الخاصية WindowState لتحديد حجم النموذج، فالنموذج يتخذ الحجم الافتراضي أو الطبيعي Normal (الحجم الافتراضي هو الحجم الذي يظهر به النموذج أثناء مرحلة التصميم)، عندما تكون قيمة الخاصية WindowState للنموذج مساوية إلى الصفر (نفس قيمة الثابت vbNormal). بينما يتخذ النموذج الحجم الأعظمي (تكبير النموذج لكامل مساحة الشاشة) عند إسناد القيمة 2 (نفس قيمة الثابت vbMaximized) للخاصية WindowState.

يُعطِّل الإجراء ()mnuSmall_Click عمل البند **صغير**، ويفعّل البند **كبير**، لأنه لا معنى أن يكون البند **صغير** فعالاً عندما يكون النموذج بحجم صغير أصلاً، وإنما يفترض أن يكون البند **كبير** هو البند الفعال لإتاحة تكبير حجم النموذج.

يتشابه عمل الإجراء ()mnuLarge_Click مع الإجراء ()mnuSmall_Click.

نص برنامج الإجراء ()mnuExit_Click

ينفذ الإجراء ()mnuExit_Click لدى اختيار البند **خروج** من قائمة **الألوان**، ويؤدي إلى إنهاء تنفيذ البرنامج:

```
Private Sub mnuLarge_Click()
    End
```

```
End Sub
```

إضافة مفاتيح الاختزال

يمكننا تحسين عمل برنامج الألوان، بإضافة مفاتيح الاختزال. تمكّنك مفاتيح الاختزال من اختيار بند قائمة ما، بالضغط على مفاتيح الاختزال المتعلقة بهذا البند من لوحة المفاتيح.

فمثلاً يمكننا إرفاق البند **أحمر** (الموجود في قائمة **الألوان**) بمفتاح الاختزال Ctrl+R، يسمح هذا العمل بالضغط على Ctrl+R لتغيير لون النموذج، بدلاً من النقر على البند **تحديد اللون** بواسطة الفأرة ثم النقر على البند **أحمر** لتغيير لون النموذج إلى الأحمر.

استخدم الخطوات التالية لإرفاق البند **أحمر** بالمفتاح الاختزالي Ctrl+R:

- اختر النموذج frmColors.
- اختر البند **Menu Editor** من قائمة **Tools**.

يستجيب فيجول بيسك بإظهار الإطار Menu Editor.

- اختر البند **أحمر** من المربع الموجود أسفل الإطار Menu Editor.

*يستجيب فيجول بيسك بإظهار خصائص البند **أحمر** في الجزء العلوي من الإطار Menu Editor.*

لاحظ أن الحقل المدعو Shortcut في الجزء العلوي، يحوي القيمة None. أي لا يوجد مفاتيح اختزال مرفقة مع هذا البند.

استخدم الخطوات التالية لتحديد Ctrl+R كمفاتيح اختزال للبند **أحمر**:

- انقر على السهم النازل الموجود يمين الحقل Shortcut في الإطار Menu Editor، ثم اختر القيمة Ctrl+R.
- انقر الزر **Ok** لإغلاق الإطار Menu Editor.

استخدم الخطوات التالية لمشاهدة مفتاح الاختزال Ctrl+R الذي أرفقته بالبند **أحمر** قيد التنفيذ:

- نفّذ برنامج الألوان.

□اضغط على المفتاحين Ctrl+R.

يستجيب برنامج الألوان بتبديل لون النموذج إلى الأحمر.

كرر الخطوات الآنفة الذكر، لإرفاق مفتاح الاختزال Ctrl+B بالبند **أزرق**، ومفتاح الاختزال Ctrl+W بالبند **أبيض**.

يدرك المستخدمون أن بوسعهم الضغط على Ctrl+R، للحصول على نفس الأثر الناتج عن اختيار البند **أحمر** من القائمة، وذلك لأن نص مفتاح الاختزال Ctrl+R يظهر على يسار البند **أحمر**. وبشكل مشابه، تظهر العبارة النصية Ctrl+B على يسار البند **أزرق**، وتظهر العبارة Ctrl+W على يسار البند **أبيض** (انظر الشكل 5-14)، ويؤدي الضغط على كل منهما من لوحة المفاتيح إلى اختيار البند المرافق له.



الشكل 5-14

مفاتيح الاختزالات Shortcuts تظهر على يسار بنود القائمة **تحديد اللون**.

ملاحظة

لا يوفر فيجول بيسك (فعلياً، ولا النظام ويندوز نفسه) مفاتيح اختزال عربية، لذلك لا يمكننا إسناد مفتاح الاختزال Ctrl+م مثلاً، إلى بند القائمة **أحمر**.

إضافة خط فاصل بين بنود القائمة

يعتبر الخط الفاصل Separator Bar عبارة عن خط أفقي يفصل بين بنود القائمة الواحدة. والغاية منه هي إعطاء منظر جمالي للقائمة. يوضح الشكل 5-15 قائمة **الألوان** بعدما أضيف إليها خط فاصل بين بند **تحديد اللون** والبند **خروج**.



الشكل 5-15
خط فاصل Separator Bar
في قائمة الألوان.

اتبع الخطوات التالية لإضافة خط فاصل بين بند **تحديد اللون** والبند **خروج**:

- اختر النموذج frmColors.
- اختر البند **Menu Editor** من قائمة **Tools**.

يستجيب فيجول بيسك بإظهار الإطار Menu Editor.

- ضع الإضاءة على البند **خروج**، ثم انقر الزر **Insert** في الإطار Menu Editor.

*يستجيب فيجول بيسك بإضافة سطر فارغ فوق البند **خروج**.*

- اكتب إشارة الناقص العادية "–" في الحقل النصي Caption من الإطار Menu Editor.
- اكتب munSep1 في الحقل النصي Name من الإطار Menu Editor.
- انقر الزر **Ok** من الإطار Menu Editor لإغلاقها.

لاحظ أن رمز إشارة الناقص العادية " – " الذي كتبته في الحقل Caption يمثل رمز الخط الفاصل Separator Bar.

لاحظ أيضاً أنك أطلقت على الخط الفاصل الاسم mnuSep1، فإذا أردت إضافة المزيد من الخطوط الفاصلة، فسوف يكون بمقدورك تسميتها mnuSep2، mnuSep3 ... الخ.

طبعاً تستطيع إطلاق أي أسماء أخرى تريدها على الخطوط الفاصلة، لكننا نحاول دائماً استخدام أسماء تسمح للمرء تمييز مهمة أي كائن برمجي من خلال اسمه.

لا يمكننا اختيار الخط الفاصل أثناء تنفيذ البرنامج، ولهذا فإن الإجراء mnuSep1_Click() لن يُنفذ أبداً. (السؤال الذي قد يتبادر إلى الذهن، ما هي الحاجة إلى تسمية بند قائمة لن ينفذ أبداً ؟. الجواب: يتطلب فيجول بيسك تسمية كل بنود القائمة مهما كان نوعها).

- نفّذ برنامج الألوان.

كما تشاهد لقد ظهر الخط الفاصل الذي أضفته للتو.

كيفية جعل أحد عناصر تحكم القائمة غير مرئي

لقد كانت عملية تفعيل أو تعطيل أحد بنود القائمة، تتم بكتابة التعليمة التي تسند إحدى القيمتين True أو False إلى الخاصية Enabled التابعة لأحد عناصر تحكم القائمة.

يظهر البند الممثل لعنصر تحكم القائمة، عند تعطيله بلون باهت (عندما تكون الخاصية Enabled مساوية للقيمة False). وعندها لا يمكن اختياره، إلا أن تعطيل أحد البنود، لا يمنع من استمرار ظهوره في القائمة.

قد نحتاج في بعض البرامج إلى إخفاء أحد بنود القائمة كلياً، وليس مجرد جعله باهتاً. تستخدم الخاصية Visible لتحقيق هذا الغرض. فمثلاً لإخفاء بند قائمة يدعى mnuMyItem خلال مرحلة التنفيذ، استخدم العبارة التالية:

```
mnuMyItem.Visible = False
```

سوف يختفي البند mnuMyItem بعد تنفيذ هذه العبارة، وتزاح كل بنود القائمة الواقعة تحته إلى الأعلى، لملء الفراغ الذي تركه. (تظهر القائمة للمستخدم وكأن هذا البند غير موجود أصلاً).

يمكن جعل كل بنود القائمة غير مرئية مرة واحدة، وذلك بإسناد القيمة False للخاصية Visible التابعة لعنوان هذه القائمة. فتختفي عندها هذه القائمة، وتزاح كل عناوين القائمة الواقعة على يسارها إلى اليمين لملء الفراغ الذي تركه هذا العنوان.

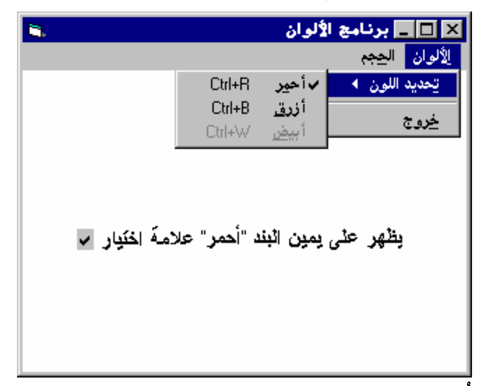
جرب كتابة التعليمة التالية في الإجراء Form_Load():

```
mnuColors.Visible = False
```

سوف يختفي عنوان قائمة **الألوان** بالإضافة لجميع بنوده، وكأن قائمة **الألوان** غير موجودة أصلاً.

استخدام علامات الاختيار Check Marks

يلزمنا في بعض البرامج وضع إشارة على أحد بنود القائمة. يمكنك استخدام علامات الاختيار لإنجاز هذه الغاية. يوضح الشكل 5-16 أحد بنود قائمة **تحديد اللون** مع علامة اختيار بجانبه.



الشكل 5-16
علامة اختيار على يسار البند أحمر.

تُستخدم الخاصية Checked لوضع أو رفع علامة الاختيار. فعند إسناد القيمة True للخاصية Checked التابعة لأحد عناصر تحكم القائمة، توضع علامة اختيار بجانب ذلك البند، أما إسناد القيمة False للخاصية Checked فيؤدي إلى إزالة علامة الاختيار من جانب هذا البند. فمثلاً استخدم العبارة التالية لوضع علامة اختيار بجانب البند **أحمر**.

```
mnuRed.Checked = True
```

أما لإزالة علامة الاختيار من جانب البند **أحمر**، فاستخدم العبارة التالية:

```
mnuRed.Checked = False
```

(لا يستخدم برنامج الألوان علامات الاختيار لبنود القائمة الموجودة فيه).

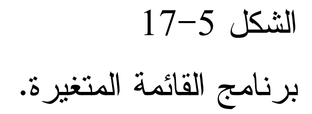
لاحظ أن أحد مربعات الاختيار التي تظهر في الجزء العلوي من الإطار Menu Editor، يدعى Checked. ووضع علامة اختيار في مربع الاختيار هذا، يعني أن بند القائمة المضاء (المنتقى) حالياً في الجزء السفلي من الإطار Menu Editor، سوف يظهر وعلى يمينه علامة اختيار (يمكنك إزالة هذه العلامة من خلال التعليمة التي تسند القيمة False للخاصية Checked التابعة لنفس البند).

برنامج القائمة المتغيرة

سنكتب الآن برنامجاً يدعى برنامج القائمة المتغيرة، حيث يوضح كيفية إضافة البنود أو إزالتها من القائمة خلال مرحلة التنفيذ. دعنا نحدد أولاً ما يفترض أن يقوم به برنامج القائمة المتغيرة، ثم نتابع بعد ذلك كتابة البرنامج.

يظهر عند بدء التشغيل شريط قائمة يتضمن العنوان **تغيير**، حسب ما يبينه الشكل 5-17.

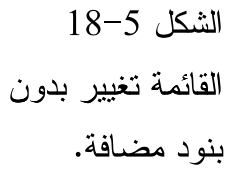



الشكل 5-17

برنامج القائمة المتغيرة.

تحتوي القائمة **تغيير** في بداية التشغيل على ثلاثة بنود هي على التوالي: **إضافة بند**، و **إزالة بند**، و **خروج**، وخط فاصل. (انظر الشكل 5-18). لاحظ أن بند القائمة **إزالة**، يظهر بلون باهت.




الشكل 5-18

القائمة تغيير بدون بنود مضافة.

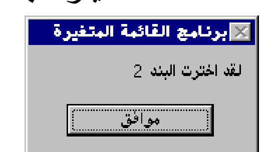
يتم إضافة بند جديد إلى القائمة **تغيير**، كلما اخترت البند **إضافة بند**. فمثلاً يوضح الشكل 5-19كيف تبدو القائمة **تغيير** بعد اختيار **إضافة بند** ثلاث مرات.



الشكل 5−19
القائمة تغيير مع
ثلاثة عناصر مضافة.

يتم حذف البند الأخير من القائمة **تغيير** كلما اخترت البند **إزالة بند**.

يظهر عند اختيار أي من العناصر التي أضفتها، رسالة تحمل اسم البند الذي اخترته. فمثلاً عند اختيار **البند 2**، تظهر الرسالة المبينة بالشكل 5−20.



الشكل5−20
الرسالة التي تظهر بعد اختيار
البند 2 من القائمة تغيير.

التصميم المرئي لبرنامج القائمة المتغيرة

نبدأ كعادتنا، بتصميم نموذج البرنامج:

- ابدأ مشروعاً من النوع التنفيذي القياسي Standard EXE.
- احفظ نموذج المشروع باسم Grow.Frm في الدليل C:\VB5Prg\Ch05، واحفظ ملف المشروع باسم Grow.Vbp في ذات الدليل.
- أنشئ النموذج وفق الجدول 5−3.

الجدول 5-3. جدول خصائص النموذج frmGrow.

الكائن	الخاصية	القيمة
Form	**Name**	**FrmGrow**
	Caption	برنامج القائمة المتغيرة
	RightToLeft	True
	BackColor	White

إنشاء قائمة برنامج القائمة المتغيرة

سنبدأ الآن بربط القائمة مع النموذج frmGrow:

- اختر النموذج frmGrow.
- اختر البند **Menu Editor** من قائمة **Tools**.

يستجيب فيجول بيسك بإظهار الإطار Menu Editor.

- صمم القائمة وفق الجدول 5-4.

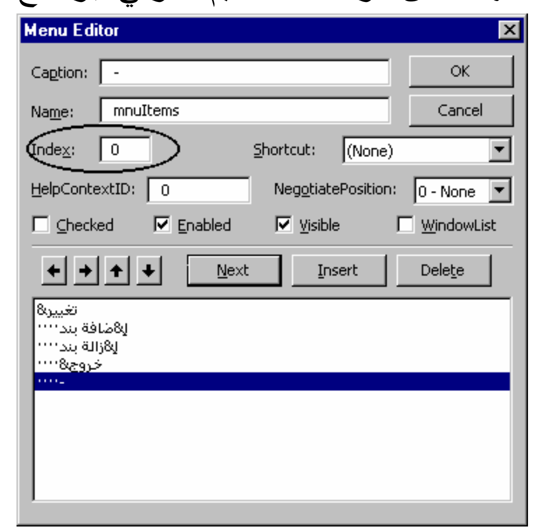
لاحظ أن البند **إضافة بند** يمتلك ثلاث نقاط قبله، مما يعني أنه مزاح يمينياً داخل الإطار Menu Editor، لاحظ أيضاً أن البنود **إزالة بند** و **خروج** والخط الفاصل (-) مزاحين أيضاً إلى اليمين في الإطار Menu Editor، مثلما كانت الحال عليه مع البنود **تحديد اللون** و **خروج** و **صغير** و **كبير**، المبينين في الشكل 5-9.

الجدول 5-4. جدول قائمة النموذج frmGrow.

الاسم	العنوان
MnuGrow	&تغيير
MnuAdd	...إ&ضافة بند
MnuRemove	...إ&زالة بند
MnuExit	...&خروج
mnuSep1	...-

- ضع الإضاءة فوق آخر بند من القائمة (الخط الفاصل)، ثم اكتب القيمة صفر في مربع النص Index (انظر الشكل 5-21).
- انقر الزر **Ok** في الإطار Menu Editor.

انتهت الآن مرحلة التصميم المرئي لبرنامج القائمة المتغيرة.



الشكل 5-21

إسناد القيمة صفر للخاصية Index التابعة للخط الفاصل.

إنشاء مصفوفة عناصر تحكم القائمة

لقد أدى إسناد القيمة صفر للخاصية Index التابعة للخط الفاصل، إلى ما يدعى بمصفوفة عناصر تحكم القائمة. سوف تدعى هذه المصفوفة بالاسم mnuItems، لأنك كتبت الاسم mnuItems في الخاصية Name.

تتألف المصفوفة mnuItems حتى هذه اللحظة، من بند واحد هو الخط الفاصل Separator Bar (أي بمعنى أن البند mnuItems(0) هو بند الخط الفاصل).

سيضيف برنامجنا مع كل اختيار للبند **إضافة بند** من القائمة **تغيير**، أثناء تنفيذ البرنامج بنداً جديداً إلى المصفوفة mnuItems. ونتيجة لذلك، سيظهر المزيد من بنود القائمة تحت بند الخط الفاصل.

إدخال نص برنامج القائمة المتغيرة

سندخل الآن نص برنامج القائمة المتغيرة كالتالي:

❑اكتب النص التالي في قسم التصاريح العامة للنموذج frmGrow:

```
'يجب التصريح عن كل المتحولات في البرنامج
Option Explicit

'gLastElement التصريح عن المتحول
Dim gLastElement As Integer
```

ملاحظة

إذا واجهت صعوبة في إضافة نص في قسم التصاريح العامة، جرب ما يلي: ضع مؤشر الفأرة عند بداية التعليمة Option Explicit ثم اضغط المفتاح Enter لعدة مرات. سيضاف نتيجة لذلك عدة أسطر فارغة إلى قسم التصاريح العامة، تستطيع الآن إدخال النص الإضافي في قسم التصاريح العامة هذا.

❑اكتب النص التالي في الإجراء Form_Load():

```
Private Sub Form_Load()
    'gLastElement إعطاء قيمة ابتدائية للمتحول
    gLastElement = 0

    'عند تشغيل البرنامج لأول مرة لا يوجد أية بنود
    'يمكن إزالتها لذلك يتم تعطيل البند إزالة
    mnuRemove.Enabled = False
End Sub
```

❑اكتب النص التالي في الإجراء mnuAdd_Click():

```
Private Sub mnuAdd_Click()
    'gLastElement زيادة المتحول
    gLastElement = gLastElement + 1

    'mnuItems[] إضافة عنصر جديد للمصفوفة
    Load mnuItems(gLastElement)
     'تحديد العنوان للبند الجديد
    mnuItems(gLastElement).Caption = _
              "البند " + Str(gLastElement)

    'بما أنه تم إضافة بند جديد للقائمة المتغيرة
```

```
    'لذلك يجب تفعيل البند إزالة بند
    mnuRemove.Enabled = True
End Sub
```

❑ اكتب النص التالي في الإجراء ()mnuRemove_Click:

```
Private Sub mnuRemove_Click()
    'mnuItems إزالة آخر بند من المصفوفة
    Unload mnuItems(gLastElement)

    'gLastElement إنقاص قيمة المتحول
    gLastElement = gLastElement - 1

    'عندما لا يتبقى في المصفوفة أية بنود
    'يجب عندها تعطيل البند إزالة
    If gLastElement = 0 Then
      mnuRemove.Enabled = False
    End If
End Sub
```

❑ اكتب النص التالي في الإجراء ()mnuItems_Click:

```
Private Sub mnuItems_Click(Index As Integer)
    'التصريح عن المتحول الذي سيحوي الرسالة
    Dim myMsg As String

    'التصريح عن المتحول الذي سيحوي عنوان الرسالة
    Dim myTitle As String
      'عرض الرسالة التي تبين البند الذي اختاره المستخدم
    myTitle = "برنامج القائمة المتغيرة"
    myMsg = "لقد اخترت البند " + Str(Index)
    MsgBox myMsg, vbOKOnly + vbMsgBoxRtlReading + _
    vbMsgBoxRight, myTitle
End Sub
```

إذا أردت الوصول إلى الإجراء ()mnuItems_Click، أظهر إطار نص البرنامج، ثم اختر من مربع السرد اليساري العلوي البند mnuItems، واختر من مربع السرد اليميني العلوي البند Click.

❑اكتب النص التالي في الإجراء ()mnuExit_Click:

```
Private Sub mnuExit_Click()
```

```
    End
End Sub
```

□احفظ المشروع الآن باختيار البند **Save Project** من قائمة **File**.

تنفيذ برنامج القائمة المتغيرة

نفّذ البرنامج ثم اختر البنود المختلفة من القائمة **تغيير**، لاحظ الظواهر التالية أثناء تنفيذ البرنامج:

يظهر البند **إزالة بند** بلون باهت عند بدء تشغيل البرنامج (لا يكون هذا البند متاحاً في البداية، لأنه لا توجد بنود مضافة حتى هذه اللحظة، يمكنك حذفها).

يضاف لدى كل اختيار للبند **إضافة بند**، بند جديد إلى القائمة تغيير، باسم (بند ×). حيث × هو رقم تسلسلي يعطى لاسم البند الجديد.

يصبح البند **إزالة بند** فعالاً حالما يضاف بند واحد على الأقل إلى القائمة **تغيير**.

يؤدي اختيار البند **إزالة بند** في كل مرة، إلى إزالة آخر بند مضاف إلى القائمة، ويصبح هذا البند (أي **إزالة بند**) معطلاً عندما تخلو القائمة من جميع البنود المضافة. (تم حذفها جميعاً).

□اختر البند **خروج** من القائمة، لإنهاء عمل البرنامج.

كيف يعمل برنامج القائمة المتغيرة

يستخدم برنامج القائمة المتغيرة المتحول gLastElement للاحتفاظ بعدد البنود في المصفوفة mnuItems. ولأن هذا المتحول يجب أن يكون مرئياً (متاحاً) لكل الإجراءات في النموذج، فقد صرحنا عنه في قسم التصاريح العامة للنموذج frmGrow:

```
'يجب التصريح عن كل المتحولات في البرنامج
Option Explicit

'gLastElement التصريح عن المتحول
Dim gLastElement As Integer
```

نص الإجراء Form_Load()

يُنفذ الإجراء Form_Load() آلياً، عند بداية تشغيل البرنامج. لاحظ أن المتحول gLastElement في هذا الإجراء، قد تم إسناد القيمة الابتدائية صفر له، كما أن البند **إزالة بند** قد أُبطِل عمله:

```
Private Sub Form_Load()
    'gLastElementإعطاء قيمة ابتدائية للمتحول
    gLastElement = 0

    'عند تشغيل البرنامج لأول مرة لا يوجد أية بنود
    'يمكن إزالتها لذلك يتم تعطيل البند إزالة
    mnuRemove.Enabled = False
End Sub
```

تعمل التعليمة الأولى في هذا الإجراء على إسناد القيمة الابتدائية صفر، للمتحول gLastElement. لأن المصفوفة mnuItems تمتلك بنداً واحداً فقط، وهو البند ذي الرقم صفر الذي تم تشكيله أثناء مرحلة التصميم (الخط الفاصل).

أما التعليمة الثانية في هذا الإجراء، فإنها تعمل على تعطيل البند **إزالة بند**، لأنه لا يوجد بند يمكن إزالته عند بداية تشغيل البرنامج.

نص الإجراء mnuAdd_Click()

ينفذ الإجراء mnuAdd_Click() عند اختيار البند **إضافة بند** من قائمة **تغيير**. وهذا الإجراء مسؤول عن إضافة بند جديد إلى المصفوفة mnuItems وإسناد عنوان مناسب للخاصية Caption التابعة لهذا البند:

```
Private Sub mnuAdd_Click()
    'gLastElement زيادة المتحول
    gLastElement = gLastElement + 1

    'mnuItems[] إضافة عنصر جديد للمصفوفة
    Load mnuItems(gLastElement)

    'تحديد العنوان للبند الجديد
```

```
    mnuItems(gLastElement).Caption = _
              "البند " + Str(gLastElement)

    'بما أنه تم إضافة بند جديد للقائمة المتغيرة'
    'لذلك يجب تفعيل البند إزالة بند'
    mnuRemove.Enabled = True
End Sub
```

أول شيء فعله هذا الإجراء كان زيادة قيمة المتحول gLastElement بمقدار 1:

```
    gLastElement = gLastElement + 1
```

ثم أضاف بنداً جديداً إلى المصفوفة mnuItems بواسطة تعليمة Load:

```
    Load mnuItems(gLastElement)
```

كما أسند عنواناً مناسباً للخاصية Caption التابعة للبند الجديد بواسطة التعليمة التالية:

```
    mnuItems(gLastElement).Caption = _
              "البند " + Str(gLastElement)
```

وأخيراً تم تفعيل البند **إزالة بند**:

```
    mnuRemove.Enabled = True
```

هذه الخطوة الأخيرة سببها أن الإجراء أضاف للتو بنداً إلى قائمة **تغيير**، ولهذا يفترض أن يكون المستخدم قادراً على إزالته.

نص الإجراء mnuRemove_Click()

ينفذ الإجراء mnuRemove_Click() عند اختيار البند **إزالة بند** من القائمة **تغيير**:

```
Private Sub mnuRemove_Click()
    'إزالة آخر بند من المصفوفة mnuItems'
    Unload mnuItems(gLastElement)

    'إنقاص قيمة المتحول gLastElement'
    gLastElement = gLastElement - 1

    'عندما لا يتبقى في المصفوفة أية بنود'
    'يجب عندها تعطيل البند إزالة'
    If gLastElement = 0 Then
      mnuRemove.Enabled = False
    End If
End Sub
```

يحذف هذا الإجراء آخر عنصر في المصفوفة mnuItems بواسطة التعليمة Unload:

```
Unload mnuItems(gLastElement)
```

يلي ذلك إنقاص قيمة المتحول gLastElement بمقدار 1. ثم يتم التحقق إذا كان البند المتبقي هو البند ذي الرقم صفر، فإذا تحقق الشرط فهذا يعني وجوب تعطيل البند **إزالة بند**:

```
If gLastElement = 0 Then
  mnuRemove.Enabled = False
End If
```

نص الإجراء mnuItems_Click()

ينفذ الإجراء mnuItems_Click() عند اختيار أي من البنود المضافة الجديدة الموجودة في القائمة تغيير:

```
Private Sub mnuItems_Click(Index As Integer)
    'التصريح عن المتحول الذي سيحوي الرسالة
    Dim myMsg As String

    'التصريح عن المتحول الذي سيحوي عنوان الرسالة
    Dim myTitle As String

    'عرض الرسالة التي تبين البند الذي اختاره المستخدم
    myTitle = "برنامج القائمة المتغيرة"
    myMsg = "لقد اخترت البند " + Str(Index)
    MsgBox myMsg, vbOKOnly + vbMsgBoxRtlReading + _
    vbMsgBoxRight, myTitle
End Sub
```

يستشعر الوسيط Index في هذا الإجراء، بالبند الذي تم اختياره. فمثلاً يساوي الوسيط Index إلى القيمة 1 عند اختيار البند المضاف الأول **بند 1**. باعتبار أن **البند 1** هو البند ذي القيمة 1 في المصفوفة mnuItems.

كما يساوي هذا الوسيط (أي Index) إلى القيمة 2 لدى اختيار **البند 2**، وهكذا تتغير قيمة الوسيط Index تلقائياً حسب البند المختار.

الخلاصة

تعلمت في هذا الفصل كيف تكتب برنامجاً يحتوي على قائمة، وكيف تربط هذه القائمة بالنموذج، باستخدام الإطار Menu Editor، وكيف تربط عناصر تحكم القائمة بنصوص البرنامج الخاصة بها.

كما تعلمت عن الخصائص المختلفة التى تناولها الفصل، مثلاً Checked, Visible التى تعطيك القدرة والمرونة على تبديل المظهر الذى تبدو به القائمة خلال مرحلة التنفيذ. ولا تنس أيضاً مصفوفة عناصر تحكم القائمة، حيث أوضح هذا الفصل آلية إنشائها وكيف تتمكن بواسطتها من إضافة بنود إلى القائمة أو حذفها وذلك خلال مرحلة التنفيذ.

الفصل السادس

مربعات الحوار

يوضح هذا الفصل كيفية دمج واستخدام مربعات الحوار في برنامجك، تستخدم مربعات الحوار لإظهار والحصول على المعلومات من المستخدم. توجد ثلاثة أنواع من مربعات الحوار في فيجول بيسك، وهذه الأنواع هي:

مربعات الحوار مسبقة التعريف Predefined Dialog Boxes أو (مربعات الحوار الجاهزة).

مربعات الحوار المخصصة Custom Dialog Boxes.

مربعات الحوار الشائعة Common Dialog Boxes.

مربعات الحوار مسبقة التعريف

حسب ما يتبدى من الاسم، فمربعات الحوار مسبقة التعريف، عبارة عن مربعات معرفة مسبقاً بواسطة فيجول بيسك. ولإظهار مربع مسبق التعريف، تستخدم عبارة ذات وسائط معينة تحدد كيف ومتى يتوجب إظهار المربع الحواري. يمكنك إظهار مربع حواري مسبق التعريف بواسطة ما يلي:

- عبارة MsgBox والتابع ()MsgBox.
- التابع الوظيفي ()InputBox.

برنامج الرسالة

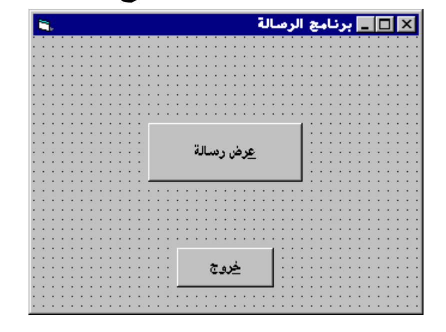
يسمح استخدام العبارة MsgBox والتابع الوظيفي ()MsgBox بإظهار الرسائل للمستخدم والحصول على استجابته (الاستجابة تكون إما نعم Yes أو لا No). يوضح برنامج الرسالة كيف تستخدم العبارة MsgBox والتابع الوظيفي ()MsgBox في البرنامج.

التمثيل المرئي لبرنامج الرسالة

سنبدأ كعادتنا بالتمثيل المرئي لنموذج البرنامج:

- أنشئ الدليل C:\VB5Prg\Ch06.
- ابدأ مشروعاً من النوع التنفيذي القياسي Standard EXE.
- احفظ نموذج المشروع بالاسم Message.Frm في الدليل C:\VB5Prg\Ch06، واحفظ ملف المشروع بالاسم Message.Vbp في الدليل ذاته.
- أنشئ النموذج تبعاً للجدول 6-1.

يفترض أن يظهر النموذج المكتمل مثل المبين في الشكل 6-.



الشكل 6-1
النموذج frmMessage
(طور التصميم).

الجدول 6−1. جدول خواص النموذج frmMessage.

القيمة	الخاصية	الكائن
FrmMessage	**Name**	**Form**
برنامج الرسالة	Caption	
True	RightToLeft	
cmdExit	**Name**	**CommandButton**
&خروج	Caption	
True	RightToLeft	
cmdMessage	**Name**	**CommandButton**
إ&ظهار رسالة	Caption	
True	RightToLeft	

إدخال نص برنامج الرسالة

سندخل الآن نص برنامج الرسالة:

- تحقق أن قسم التصاريح العامة من النموذج frmMessage يحوي العبارة Option Explicit، أي:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء cmdMessage_Click():

```
Private Sub cmdMessage_Click()
    Dim Message As String
    Dim Title   As String
    Dim ButtonsAndIcon As Long

    'تحديد الرسالة التي ستظهر في المربع الحواري
    Message = "هذا مثال على رسالة بسيطة"
```

```
    'تحديد نوع الأزرار المفترض ظهورها والصورة أيضا
    ButtonsAndIcon = vbOKOnly + vbExclamation + _
                     vbMsgBoxRight + vbMsgBoxRtlReading

    'تحديد العنوان
    Title = "عنوان المربع الحواري"

    'إظهار المربع الحواري
    MsgBox Message, ButtonsAndIcon, Title

End Sub
```

▫ أدخل النص التالي ضمن الإجراء cmdExit _Click():

```
Private Sub cmdExit_Click()
    Dim Message As String
    Dim Title   As String
    Dim ButtonsAndIcon As Long
    Dim Response       As String

    'تحديد الرسالة التي ستظهر في المربع الحواري
    Message = "هل أنت متأكد بأنك تريد الخروج؟"

    'تحديد نوع الأزرار المفترض ظهورها والصورة أيضا
    ButtonsAndIcon = vbYesNo + vbQuestion + _
               vbMsgBoxRight + vbMsgBoxRtlReading

    'تحديد العنوان
    Title = "برنامج الرسالة"

    'إظهار المربع الحواري وانتظار رد المستخدم
    Response = MsgBox(Message, ButtonsAndIcon, Title)

    'تقييم استجابة المستخدم
    If Response = vbYes Then
      End
    End If

End Sub
```

▫ اختر البند **Save Project** من القائمة **File** لفيجول بيسك لحفظ عملك.

تنفيذ برنامج الرسالة

دعنا ننفذ برنامج الرسالة، ونرى نتيجة ما كتبناه على أرض الواقع.

□نفّذ برنامج الرسالة.

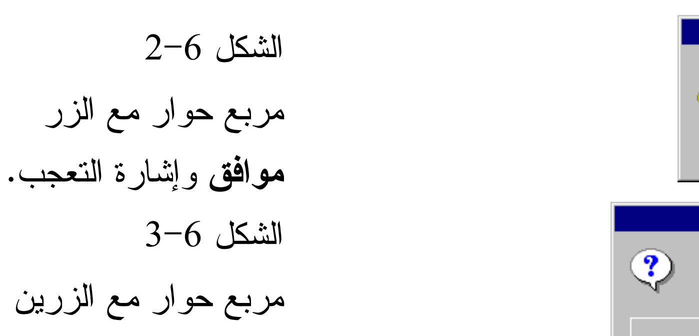
لاحظ الظاهرة التالية أثناء تنفيذ البرنامج: يظهر مربع حوار مع رسالة عندما تضغط على الزر ***عرض رسالة****، كما يحمل المربع الزر* ***موافق*** *ورمز* ***إشارة التعجب*** *Exclamation Icon، انظر الشكل 6-2. مجال هذا المربع الحواري محدد (أو محصور) بمعنى أن البرنامج لا يستطيع استئناف التنفيذ ما لم تغلق مربع الحوار هذا. فمثلاً جرب النقر بواسطة الفأرة على الزر* ***خروج*** *أثناء ظهور مربع الحوار هذا، وستجد أن البرنامج لن ينتهي (أي لم ينفذ الإجراء cmdExit_Click()).*

□انقر على الزر **موافق** في مربع الحوار.

يغلق مربع الحوار ذاته، استجابة لذلك.

□انقر على الزر **خروج**.

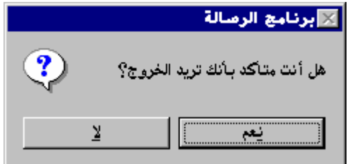
يظهر نتيجة لذلك مربع حواري يحمل زراً يدعى **نعم** وآخر **لا**، ورمز إشارة الاستفهام (انظر الشكل 6-3). يطلب منك هذا المربع تثبيت رغبتك في الخروج. يؤدي الضغط على **نعم** إلى إنهاء البرنامج، أما الضغط على الزر **لا** فيؤدي إلى إغلاق المربع الحواري ولكن بدون انتهاء البرنامج. يعتبر مربع الحوار هذا أيضاً محدداً (مشروطاً Modal) بمعنى أنك لا تستطيع استئناف البرنامج ما لم تستجب لهذا المربع بالنقر على أحد الزرين **نعم** أو **لا**.





الشكل 6-2

مربع حوار مع الزر **موافق** وإشارة التعجب.



الشكل 6-3

مربع حوار مع الزرين **نعم** و **لا** و إشارة استفهام.

ملاحظة

تدعى مربعات الحوار التي تناولها مثالنا هذا، مربعات حوار محددة Modal Dialog Boxes. يمكن أثناء عرض مربع الحوار من هذا النوع (مثل مربع الرسالة) الانتقال إلى برامج ويندوز الأخرى، لكن ما لا يمكن هو، العودة إلى نافذة برنامج الرسالة دون إغلاق مربع الحوار هذا.

نوع آخر من مربعات الحوار المحددة، هو مربع حوار النظام المحدد، الذي لا يسمح لك بالرجوع إلى التطبيق، ولا يمكّنك من الانتقال إلى تطبيقات ويندوز أخرى. أي بمعنى آخر، لا تستطيع العمل خارج هذا المربع حتى تجيب عليه.

كيف يعمل برنامج الرسالة؟

يستخدم برنامج الرسالة العبارة MsgBox والتابع الوظيفي MsgBox() لإظهار مربعات حوار الرسائل.

إظهار مربع حوار بواسطة العبارة MsgBox

ينفذ الإجراء cmdMessage_Click() آلياً عند الضغط على الزر **عرض رسالة**. يستخدم البرنامج المكتوب داخل هذا الإجراء العبارة MsgBox لإظهار مربع حوار (مربع رسالة) يحمل زراً يدعى **موافق** ورمز لإشارة التعجب:

```
Private Sub cmdMessage_Click()

    Dim Message As String
    Dim Title   As String
    Dim ButtonsAndIcon As Long

    'تحديد الرسالة التي ستظهر في المربع الحواري
    Message = "هذا مثال على رسالة بسيطة"

    'تحديد نوع الأزرار المفترض ظهورها والصورة أيضا
    ButtonsAndIcon = vbOKOnly + vbExclamation + _
                     vbMsgBoxRight + vbMsgBoxRtlReading
```

```
'تحديد العنوان'
Title = "عنوان المربع الحواري"

'إظهار المربع الحواري'
MsgBox Message, ButtonsAndIcon, Title

End Sub
```

لاحظ أننا مررنا ثلاثة وسائط إلى العبارة MsgBox:

الرسالة المراد إظهارها (سلسة كتابية String).

الأزرار والرموز المراد ظهورها في مربع الرسالة (قيمة عددية).

عنوان مربع الحوار (سلسلة كتابية String).

يجهز الإجراء cmdMessage_Click() المتحولات الثلاثة اللازمة كوسائط في عبارة MsgBox قبل استخدام هذه العبارة. وأول متحول قام الإجراء بتكليفه كما رأينا هو المتحول Message الذي يحمل الرسالة التي ستظهر في مربع الحوار:

```
Message = "هذا مثال على رسالة بسيطة"
```

يستخدم المتحول Message كأول وسيط في العبارة MsgBox.

المتحول الثاني عبارة عن متحول من النوع الصحيح Integer ويدعى ButtonsAndIcon. ويتم تكليفه بعدد يحدد نوع الأزرار وشكل الرموز التي ستظهر في مربع الرسالة، بالإضافة إلى الثابتين vbMsgBoxRight و vbMsgBoxRtlReading اللذين يقومان بتعريب مربع الرسالة:

```
ButtonsAndIcon = vbOKOnly + vbExclamation + _
                 vbMsgBoxRight + vbMsgBoxRtlReading
```

يُستخدم المتحول ButtonsAndIcon كثاني وسيط في العبارة MsgBox كما أن vbOKOnly و vbExclamatian عبارة عن ثابتين من ثوابت فيجول بيسك. يمثل الثابت vbOKOnly الزر **موافق**، بينما يمثل الثابت vbExclamatian رمز إشارة التعجب.

وإسناد مجموع الثابتين (vbOKOnly و vbExclamatian) إلى المتحول ButtonsAndIcon، يفرض على المربع الحواري أن يمتلك الزر **موافق** ورمز إشارة التعجب، حسب ما يوضحه الشـــكل 6-2.

يعطي الجدولان 6-2 و 6-3 ثوابت الأزرار وثوابت الأيقونات التي يمكنك استخدامها من أجل الوسيط الثاني في عبارة MsgBox.

الجدول 6-2. ثوابت أزرار مربع حوار الرسالة.

اسم الثابت	القيمة	الأزرار التي ستظهر
vbOKOnly	0	موافق
vbOKCancel	1	موافق و إلغاء
vbAbortReryIgnore	2	إحباط و إعادة المحاولة و تجاهل
vbYesNoCancel	3	نعم و لا و إلغاء
vbYesNo	4	نعم و لا
vbRetryCancel	5	إعادة المحاولة و إلغاء

الجدول 6-3. ثوابت رموز مربع حوار الرسالة.

اسم الثابت	القيمة	الرمز التي ستظهر
vbCirtical	16	رمز الخطأ الحرج
vbQuestion	32	رمز إشارة الاستفهام
vbExclamation	48	رمز التعجب
vbInformation	64	رمز المعلومات

المتحول الثالث الذي يعمل الإجراء cmdMessage_Click() على إعداده، هو المتحول Title، ويتم إسناد عنوان مربع الحوار إلى ذلك المتحول:

```
Title = "عنوان المربع الحواري"
```

وطبعاً يُستخدم Title كثالث وسيط في العبارة MsgBox، وبنهاية الأمر، ينتج عن استخدام الوسائط الثلاثة Messag و ButtonsAndIcon و Title في عبارة المربع الحواري ما هو مبين في الشكل 6-2.

```
MsgBox Message, ButtonsAndIcon, Title
```

إظهار مربع حوار بواسطة التابع الوظيفي MsgBox()

ينفذ الإجراء cmdExit_Click() آلياً، لدى النقر على الزر **خروج**.

يستخدم هذا الإجراء، التابع الوظيفي MsgBox() لإظهار مربع حوار يحمل إشارة استفهام وزرين هما **نعم** و **لا**:

```
Private Sub cmdExit_Click()
    Dim Message As String
    Dim Title   As String
    Dim ButtonsAndIcon As Long
    Dim Response       As String

    'تحديد الرسالة التي ستظهر في المربع الحواري
    Message = "هل أنت متأكد بأنك تريد الخروج؟"

    'تحديد نوع الأزرار المفترض ظهورها والصورة أيضا
    ButtonsAndIcon = vbYesNo + vbQuestion + _
               vbMsgBoxRight + vbMsgBoxRtlReading

    'تحديد العنوان
    Title = "برنامج الرسالة"

    'إظهار المربع الحواري وانتظار رد المستخدم
    Response = MsgBox(Message, ButtonsAndIcon, Title)

    'تقييم استجابة المستخدم
    If Response = vbYes Then
      End
    End If
End Sub
```

يأخذ التابع الوظيفي MsgBox() نفس الوسائط التي تأخذها العبارة MsgBox، والاختلاف الوحيد بينهما، أن التابع الوظيفي MsgBox() يعيد قيمةً تشير إلى الزر الذي تم النقر عليه في مربع الحوار.

يعمل الإجراء cmdExit_Click() على تكليف المتحولات الثلاثة Message و ButtonsAndIcon و Title قبل تنفيذ التابع الوظيفي MsgBox() لاستخدامها فيه:

```
Message = "هل أنت متأكد بأنك تريد الخروج؟"
ButtonsAndIcon = vbYesNo + vbQuestion + _
           vbMsgBoxRight + vbMsgBoxRtlReading
Title = "برنامج الرسالة"
```

تم إسناد مجموع الثابتين vbYesNo و vbQuestion إلى المتحول ButtonsAndIcon لأن مربع الحوار يجب أن يمتلك الزرين **نعم** و **لا** ورمز إشارة الاستفهام.

يُنفَذ التابع الوظيفي حال الانتهاء من إعداد وسائطه، ويعيد قيمةً يتم إسنادها إلى المتحول Response.

```
Response = MsgBox(Message, ButtonsAndIcon, Title)
```

تدل القيمة التي يعيدها التابع MsgBox() على الزر الذي تم النقر أو الضغط عليه. يبين الجدول 6-4، كل الثوابت المحتملة التي يتمكن التابع MsgBox() من إعادتها. فمثلاً إذا كانت القيمة المعادة من قبل التابع الوظيفي MsgBox() هي الثابت vbYes، فهذا يعني أن المستخدم قام بالنقر على الزر **نعم** في مربع الرسالة.

الجدول 6-4. القيم المحتملة المعادة من قبل التابع الوظيفي MsgBox():

اسم الثابت	القيمة	الزر الذي تم نقره
vbOK	1	الزر موافق
vbCancel	2	الزر إلغاء

الزر إحباط	3	vbAbort
الزر إعادة المحاولة	4	vbRetry
الزر تجاهل	5	vbIgnore
الزر نعم	6	vbYes
الزر لا	7	vbNo

تستخدم عبارة If مع قيمة المتحول Responce للتحقق من الزر الذي تم النقر عليه (أي للتعرف عليه):

```
If Response = vbYes Then
  End
End If
```

عند الضغط على الزر **نعم** فإن قيمة Response تساوي vbYes، وينتهي البرنامج نتيجة لتنفيذ عبارة **End**. وهكذا تكون قد أجبت على السؤال "هل أنت متأكد بأنك تريد الخروج؟"، بالموافقة، وبذلك ينتهي البرنامج.

ملاحظة

تنفذ العبارة MsgBox في الإجراء cmdMessage_Click() كالتالي:

MsgBox Message , ButtonsAndIcon , Title

وينفذ التابع الوظيفي في الإجراء cmdExit_Click() كالتالي:

Response = MsgBox(Message , DialogType , Title)

فعند استخدام العبارة MsgBox لا يتم إحاطة الوسائط بقوسين، بعكس التابع الوظيفي MsgBox() الذي يتوجب فيه إغلاق الوسائط بقوسين، ولا تنسى وجوب استخدام القيمة المعادة من قبل التابع الوظيفي لإسنادها إلى متحول.

برنامج مربعات الحوار

أوضح برنامج الرسالة السابق، كيفية استخدام العبارة MsgBox والتابع الوظيفي MsgBox() لإظهار مربع حواري يحمل الزر **موافق**، ومربع حواري يحمل الزرين **نعم** و **لا**.

تستطيع حسب ما يوضحه الجدول 6-2 إظهار أزرار قياسية أخرى في مربعات الحوار، بالاستعانة بالعبارة والتابع الوظيفي المذكورين.

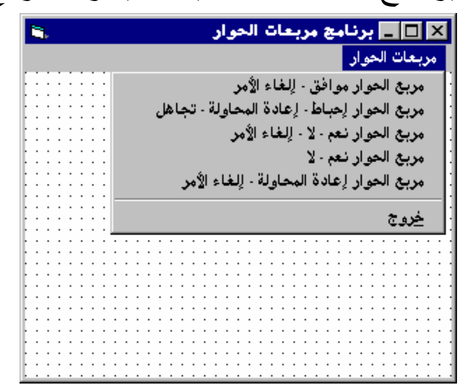
يوضح برنامج مربعات الحوار كيفية استخدام التابع الوظيفي MsgBox() لإظهار مختلف أنواع الأزرار وكيفية معرفة القيمة المعادة من التابع MsgBox() لتحديد الزر الذي تم النقر عليه.

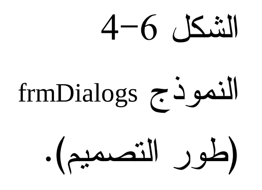
التمثيل المرئي لبرنامج مربعات الحوار

كعادتنا سنبدأ بخطوة التمثيل المرئي للنموذج frmDialogs.

- ابدأ مشروعاً قياسياً جديداً Standard EXE.
- احفظ نموذج المشروع بالاسم Dialogs.Frm في الدليل C:\VB5Prg\Ch06، ثم احفظ ملف المشروع في الدليل ذاته، بالاسم Dialoge.VBp.
- أنشئ النموذج وفقاً للجدول 6-5 والجدول 6-6 (الجدول 6-6 هوالجدول المستخدم لبناء قائمة النموذج frmDialogs).

يوضح الشكل 6-4 كيف سيبدو النموذج بعد اكتماله.




الشكل 6-4
النموذج frmDialogs
(طور التصميم).

الجدول 6-5. جدول خصائص النموذج frmDilogs.

الكائن	الخاصية	**القيمة**
Form	**Name**	**frmDialogs**
	Caption	برنامج مربعات الحوار
	RightToLeft	True

	BackColor	White
Menu	(انظر الجدول 6-6)	(انظر الجدول 6-6)

الجدول 6-6. جدول قائمة النموذج frmDilogs.

العنوان	الخاصية Name
&مربعات الحوار	mnuDialogs
...مربع الحوار موافق - إلغاء الأمر	mnuOKCancel
...مربع الحوار إحباط - إعادة المحاولة - تجاهل	mnuAbortRetryIgnore
...مربع الحوار نعم - لا - إلغاء الأمر	mnuYesNoCancel
...مربع الحوار نعم - لا	mnuYesNo
...مربع الحوار إعادة المحاولة - إلغاء الأمر	mnuRetryCancel
...-	mnuSep1
...&خروج	mnuExit

إدخال نص برنامج مربعات الحوار

سندخل الآن نص برنامج مربعات الحوار:

- تحقق بأن العبارة Option Explicit تقع في قسم التصاريح العامة للنموذج frmDialogs:

```
'All Variables Must Be Declared
Option Explicit
```

- أدخل النص التالي ضمن الإجراء mnuAbortRetryIgnore_Click():

```
Private Sub mnuAbortRetryIgnore_Click()
    Dim DialogType  As Long
    Dim DialogTitle As String
    Dim DialogMsg   As String
    Dim Response    As Integer
    'تحديد الأزرار الواجب إظهارها
    DialogType = vbAbortRetryIgnore + vbExclamation
```

```
    'تحديد الصفات العربية لمربع الحوار
    DialogType = DialogType + vbMsgBoxRight + _
                 vbMsgBoxRtlReading
    'تحديد عنوان مربع الحوار
    DialogTitle = "عنوان مربع الحوار"
    'تحديد رسالة مربع الحوار
    DialogMsg = "هذا مثال على مربع حوار يظهر الأزرار التالية:"
    'إظهار مربع الحوار والحصول على رد المستخدم
    Response = MsgBox(DialogMsg, DialogType, DialogTitle)

    'تقييم رد المستخدم
    Select Case Response
      Case vbAbort
        MsgBox "لقد نقرت الزر إحباط"
      Case vbRetry
        MsgBox "لقد نقرت الزر إعادة المحاولة"
      Case vbIgnore
        MsgBox "لقد نقرت الزر تجاهل"
    End Select
End Sub
```

- أدخل النص التالي ضمن الإجراء ()mnuExit_Click:

```
Private Sub mnuExit_Click()
    Dim DialogType  As Long
    Dim DialogTitle As String
    Dim DialogMsg   As String
    Dim Response    As Integer
    'تحديد الأزرار الواجب إظهارها
    DialogType = vbYesNo + vbCritical
    'تحديد الصفات العربية لمربع الحوار
    DialogType = DialogType + vbMsgBoxRight + vbMsgBoxRtlReading
    'تحديد عنوان مربع الحوار
    DialogTitle = "برنامج مربعات الحوار"
    'تحديد رسالة مربع الحوار
    DialogMsg = "هل أنت متأكد بأنك تريد الخروج؟"
    'إظهار مربع الحوار والحصول على رد المستخدم
    Response = MsgBox(DialogMsg, DialogType, DialogTitle)
```

```
    'تقييم رد المستخدم
    If Response = vbYes Then
      End
    End If
End Sub
```

- أدخل النص التالي ضمن الإجراء ()mnuOKCancel_Click:

```
Private Sub mnuOKCancel_Click()
    Dim DialogType  As Long
    Dim DialogTitle As String
    Dim DialogMsg   As String
    Dim Response    As Integer
    'تحديد الأزرار الواجب إظهارها
    DialogType = vbOKCancel+ vbExclamation
    'تحديد الصفات العربية لمربع الحوار
    DialogType = DialogType + vbMsgBoxRight + vbMsgBoxRtlReading
    'تحديد عنوان مربع الحوار
    DialogTitle = "عنوان مربع الحوار"
    'تحديد رسالة مربع الحوار
    DialogMsg = "هذا مثال على مربع حوار يظهر الأزرار التالية:"
    'إظهار مربع الحوار والحصول على رد المستخدم
    Response = MsgBox(DialogMsg, DialogType, DialogTitle)

    'تقييم رد المستخدم
    Select Case Response
      Case vbOK
        MsgBox "لقد نقرت الزر موافق"
      Case vbCancel
        MsgBox "لقد نقرت الزر إلغاء الأمر"
    End Select
End Sub
```

- أدخل النص التالي ضمن الإجراء ()mnuRetryCancel_Click:

```
Private Sub mnuRetryCancel_Click()
    Dim DialogType  As Long
    Dim DialogTitle As String
    Dim DialogMsg   As String
    Dim Response    As Integer
```

```
    'تحديد الأزرار الواجب إظهارها
    DialogType = vbRetryCancel+ vbExclamation
    'تحديد الصفات العربية لمربع الحوار
    DialogType = DialogType + vbMsgBoxRight + vbMsgBoxRtlReading
    'تحديد عنوان مربع الحوار
    DialogTitle = "عنوان مربع الحوار"
    'تحديد رسالة مربع الحوار
    DialogMsg = "هذا مثال على مربع حوار يظهر الأزرار التالية:"
    'إظهار مربع الحوار والحصول على رد المستخدم
    Response = MsgBox(DialogMsg, DialogType, DialogTitle)

    'تقييم رد المستخدم
    Select Case Response
      Case vbRetry
        MsgBox "لقد نقرت الزر إعادة المحاولة"
      Case vbCancel
        MsgBox "لقد نقرت الزر إلغاء الأمر"
    End Select
End Sub
```

- أدخل النص التالي ضمن الإجراء mnuYesNo_Click():

```
Private Sub mnuYesNo_Click()
    Dim DialogType  As Long
    Dim DialogTitle As String
    Dim DialogMsg   As String
    Dim Response    As Integer

    'تحديد الأزرار الواجب إظهارها
    DialogType = vbYesNo + vbQuestion
    'تحديد الصفات العربية لمربع الحوار
    DialogType = DialogType + vbMsgBoxRight + vbMsgBoxRtlReading
    'تحديد عنوان مربع الحوار
    DialogTitle = "عنوان مربع الحوار"
    'تحديد رسالة مربع الحوار
    DialogMsg = "هذا مثال على مربع حوار يظهر الأزرار التالية:"
    'إظهار مربع الحوار والحصول على رد المستخدم
    Response = MsgBox(DialogMsg, DialogType, DialogTitle)
```

```
    'تقييم رد المستخدم
    Select Case Response
      Case vbYes
        MsgBox "لقد نقرت الزر نعم"
      Case vbNo
        MsgBox "لقد نقرت الزر إلغاء لا"
    End Select
End Sub
```

❑ أدخل النص التالي ضمن الإجراء ()mnuYesNoCancel_Click:

```
Private Sub mnuYesNoCancel_Click()
    Dim DialogType  As Long
    Dim DialogTitle As String
    Dim DialogMsg   As String
    Dim Response    As Integer

    'تحديد الأزرار الواجب إظهارها
    DialogType = vbYesNoCancel+ vbExclamation
    'تحديد الصفات العربية لمربع الحوار
    DialogType = DialogType + vbMsgBoxRight + vbMsgBoxRtlReading
    'تحديد عنوان مربع الحوار
    DialogTitle = "عنوان مربع الحوار"
    'تحديد رسالة مربع الحوار
    DialogMsg = "هذا مثال على مربع حوار يظهر الأزرار التالية:"
    'إظهار مربع الحوار والحصول على رد المستخدم
    Response = MsgBox(DialogMsg, DialogType, DialogTitle)

    'تقييم رد المستخدم
    Select Case Response
      Case vbYes
        MsgBox "لقد نقرت الزر نعم"
      Case vbNo
        MsgBox "لقد نقرت الزر لا"
      Case vbCancel
        MsgBox "لقد نقرت الزر إلغاء الأمر"
    End Select
End Sub
```

▫احفظ المشروع باختيار البند **Save Project** من القائمة **File**.

تنفيذ برنامج مربعات الحوار

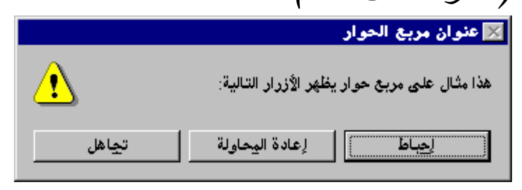
دعنا نشاهد نتيجة ما كتبناه قيد التنفيذ:

▫نفّذ برنامج مربعات الحوار.

▫اختبر وتمرن على مختلف مربعات الحوار، فمثلاً أنجز ما يلي لإظهار مربع الحوار **إحباط–إعادة المحاولة–تجاهل**:

▫اختر البند **إحباط–إعادة المحاولة–تجاهل** من قائمة مربعات الحوار.

*يستجيب البرنامج بإظهار مربع حوار يحوي الأزرار **إحباط–إعادة المحاولة–تجاهل**.* *(انظر الشكل 6-5).*



الشكل 6-5

مربع حوار يحوي الأزرار الثلاثة: **إحباط** و **إعادة المحاولة** و **تجاهل**.

استخدم الخطوتين التاليتين لإنهاء برنامج مربعات الحوار:

▫اختر البند **خروج** من قائمة مربعات الحوار.

*يستجيب البرنامج بإظهار مربع الحوار **نعم/لا**، ويتحقق من رغبتك بالخروج.*

▫انقر فوق الزر **نعم**.

يستجيب برنامج مربعات الحوار بإنهاء نفسه.

كيف يعمل برنامج مربعات الحوار

يستخدم برنامج مربعات الحوار التابع الوظيفي MsgBox() لإظهار شتى أشكال مربعات الحوار مع مختلف الأزرار. ويستخدم البرنامج القيمة المعادة من التابع الوظيفي MsgBox() لتحديد الزر الذي تم النقر عليه.

الإجراء ()mnuAbortRetryIgnore_Click

ينفذ هذا الإجراء عند اختيار البند **إحباط–إعادة المحاولة–تجاهل**، من قائمة مربعات الحوار. يستخدم نص برنامج الإجراء هذا التابع الوظيفي ()MsgBox لإظهار مربع حوار يحمل الأزرار الثلاثة: **إحباط–إعادة المحاولة–تجاهل**:

```
Private Sub mnuAbortRetryIgnore_Click()
    Dim DialogType  As Long
    Dim DialogTitle As String
    Dim DialogMsg   As String
    Dim Response    As Integer

    'تحديد الأزرار الواجب إظهارها'
    DialogType = vbAbortRetryIgnore + vbExclamation
    'تحديد الصفات العربية لمربع الحوار'
    DialogType = DialogType + vbMsgBoxRight + _
                 vbMsgBoxRtlReading
    'تحديد عنوان مربع الحوار'
    DialogTitle = "عنوان مربع الحوار"
    'تحديد رسالة مربع الحوار'
    DialogMsg = "هذا مثال على مربع حوار يظهر الأزرار التالية:"
    'إظهار مربع الحوار والحصول على رد المستخدم'
    Response = MsgBox(DialogMsg, DialogType, DialogTitle)

    'تقييم رد المستخدم'
    Select Case Response
      Case vbAbort
        MsgBox "لقد نقرت الزر إحباط"
      Case vbRetry
        MsgBox "لقد نقرت الزر إعادة المحاولة"
      Case vbIgnore
        MsgBox "لقد نقرت الزر تجاهل"
    End Select
End Sub
```

يعدل الإجراء ()mnuAbortRetryIgnore_Click المتحولات DialogType و DialogMsg و DialogTitle قبل أن ينفذ التابع الوظيفي ()MsgBox:

```
DialogType = DialogType + vbMsgBoxRight + _
             vbMsgBoxRtlReading
DialogTitle = "عنوان مربع الحوار"
DialogMsg = "هذا مثال على مربع حوار يظهر الأزرار التالية:"
Response = MsgBox(DialogMsg, DialogType, DialogTitle)
```

أسند مجموع كل من vbAbortRetryIgnore و vbExclamation إلى المتحول DialogType، لأن مربع الحوار يفترض أن يحمل الأزرار **إحباط–إعادة المحاولة–تجاهل** إلى جانب إشارة التعجب.

ينفذ التابع الوظيفي MsgBox() حال الانتهاء من إعداد المتحولات المذكورة، ويسند القيمة المعادة من التابع إلى المتحول Response:

```
Response = MsgBox(DialogMsg, DialogType, DialogTitle)
```

تختبر قيمة المتحول Response بواسطة عبارة Select Case لتحديد الزر الذي تم النقر عليه (أي أحد الأزرار الثلاثة **إحباط–إعادة المحاولة–تجاهل**)، باستخدام التعبير التالي:

```
Select Case Response
  Case vbAbort
    MsgBox "لقد نقرت الزر إحباط"
  Case vbRetry
    MsgBox "لقد نقرت الزر إعادة المحاولة"
  Case vbIgnore
    MsgBox "لقد نقرت الزر تجاهل"
End Select
```

فإذا نقرت بواسطة الفأرة على الزر **تجاهل** على سبيل المثال، فسوف تساوي قيمة المتحول Response إلى vbIgnore، وبالتالي سوف تنفذ العبارة الواقعة تحت Case vbIgnore (يوضح الجدول 6–4 معاني الثوابت vbAbort و vbRetry و vbIgnore).

تأخذ عبارات MsgBox الواردة تحت عبارة Select Case وسيطاً واحداً فقط وهو الرسالة المراد عرضها. مثال ذلك، عبارة MsgBox الواردة تحت السطر Case vbAbort، أي:

```
MsgBox "لقد نقرت الزر إحباط"
```

يؤدي عدم تحديد الوسيطين الثاني والثالث لعبارة MsgBox إلى ظهور مربع حوار يحمل زر **موافق** فقط، ودون أي رمز. كما يطلق على هذا المربع عنواناً افتراضياً وهو اسم البرنامج (اسم ملف المشروع).

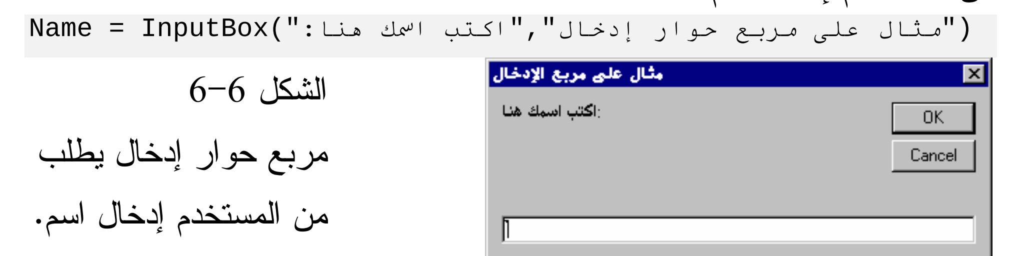
تتشابه باقي الإجراءات في هذا البرنامج مع الإجراء mnuAbortRetryIgnre_Click(). فتقوم بإعداد المتحولات DialogMsg و DialogType و DialogTitle، ثم تستخدم التابع الوظيفي MsgBox() لإظهار مربع الحوار، ثم تستخدم بعد ذلك القيمة المعادة من التابع الوظيفي MsgBox() لتحديد الزر الذي تم النقر عليه.

التابع الوظيفي InputBox

يمكن استخدام التابع الوظيفي InputBox() لتلقي معلومات كتابية من قبل المستخدم. يُظهر التابع الوظيفي InputBox() مربع حوار مع رسالة وزرين هما: **OK** و **Cancel**، ويستطيع المستخدم إدخال نص ما في الحقل النصي وإغلاق مربع الحوار بالنقر على الزر **OK**.

الوسيط الأول من التابع الوظيفي InputBox() هو رسالة مربع الحوار، والوسيط الثاني هو عنوان مربع الحوار. يعيد التابع الوظيفي InputBox() ما أدخله المستخدم في الحقل النصي. فمثلاً تظهر العبارة التالية، مربع الحوار المبين في الشكل 6–6 والذي يطلب من المستخدم إدخال اسم:

```
Name = InputBox(":اكتب اسمك هنا","مثال على مربع حوار إدخال")
```



الشكل 6–6

مربع حوار إدخال يطلب من المستخدم إدخال اسم.

يُدخل المستخدم اسماً ما في الحقل النصي. ثم ينقر الزر **OK** لإغلاق مربع الحوار. وفي مثالنا هذا نستخدم القيمة العائدة من التابع الوظيفي InputBox() لإسنادها إلى المتحول Name.

فيتم إسناد الاسم الذي أدخله المستخدم للمتحول Name فور نقر الزر **OK**.

لنحسن الآن برنامج مربعات الحوار بحيث يُستخدم التابع الوظيفي InputBox()، لتلقي سلسلة أحرف أو عدد أو تاريخ من المستخدم.

□أضف البنود الأربعة الجديدة التالية، إلى قائمة برنامج مربعات الحوار وضعها فوق بند قائمة **خروج**.

العنوان	**الخاصية Name**
...إدخال نص	mnuGetString
...إدخال رقم	mnuGetNumber
...إدخال تاريخ	mnuGetDate
...-	mnuSep2

وبعد إضافة هذه العناصر يفترض أن يكون جدول القائمة مطابقاً للجدول 6-7.

جدول 6-7. جدول قائمة البرنامج الجديدة.

العنوان	**الخاصية Name**
&مربعات الحوار	mnuDialogs
...مربع الحوار موافق - إلغاء الأمر	mnuOKCancel
...مربع الحوار إحباط - إعادة المحاولة - تجاهل	mnuAbortRetryIgnore
...مربع الحوار نعم - لا - إلغاء الأمر	mnuYesNoCancel
...مربع الحوار نعم - لا	mnuYesNo
...مربع الحوار إعادة المحاولة - إلغاء الأمر	mnuRetryCancel
...-	mnuSep1

mnuGetString	...إدخال نص
mnuGetNumber	...إدخال رقم
mnuGetDate	...إدخال تاريخ
mnuSep2	–...
mnuExit	...&خروج

- اكتب النص التالي ضمن الإجراء ()mnuGetString_Click:

```
Private Sub mnuGetString_Click()
    Dim UserInput

    'الحصول من المستخدم على إدخال نصي
    UserInput = _
      InputBox("اكتب هنا أي نص:" , "مثال على مربع الإدخال")
    'إذا لم يكتب المستخدم أي شيء أنه ضغط الزرإلغاء الأمر
    If UserInput = "" Then
      MsgBox "أنت لم تكتب أي نص أو أنك ضغطت الزر إلغاء الأمر"
      Exit Sub
    End If
    'إظهار رسالة بما كتبه المستخدم
    MsgBox "لقد كتبت ما يلي: " & UserInput
End Sub
```

- اكتب النص التالي ضمن الإجراء ()mnuGetNumber_Click:

```
Private Sub mnuGetNumber_Click()
    Dim UserInput

    'الحصول من المستخدم على إدخال رقمي
    UserInput = _
      InputBox("اكتب هنا أي رقم:" , "مثال على مربع الإدخال")
    'إذا لم يكتب المستخدم أي شيء أنه ضغط الزر إلغاء الأمر
    If UserInput = "" Then
      MsgBox "أنت لم تكتب أي شيء أو أنك ضغطت الزر إلغاء الأمر"
      Exit Sub
    End If
    'إذا لم يكن ما كتبه المستخدم رقما فاخرج من الإجراء
    If Not IsNumeric(UserInput) Then
```

```
    MsgBox "الرقم الذي كتبته غير صحيح"
    Exit Sub
  End If
  'إظهار رسالة بالرقم الذي كتبه المستخدم
  MsgBox "لقد كتبت الرقم التالي: " & UserInput
End Sub
```

❑ اكتب النص التالي ضمن الإجراء ()mnuGetData_Click:

```
Private Sub mnuGetDate_Click()
  Dim UserInput, DayOfWeek, Msg

  'الحصول من المستخدم على إدخال زمني
  UserInput = _
    InputBox("اكتب هنا أي تاريخ:" , "مثال على مربع الإدخال")
  'إذا لم يكتب المستخدم أي شيء أنه ضغط الزرإلغاء الأمر
  If UserInput = "" Then
    MsgBox "أنت لم تكتب أي شيء أو أنك ضغطت الزر إلغاء الأمر"
    Exit Sub
  End If
  'إذا لم يكن ما كتبه المستخدم تاريخا فاخرج من الإجراء
  If Not IsDate(UserInput) Then
    MsgBox "التاريخ الذي كتبته غير صحيح"
    Exit Sub
  End If

  DayOfWeek = Format(UserInput, "dddd")
  'إظهار رسالة بالتاريخ الذي أدخله المستخدم
  'وإظهار النتيجة أيضا
  Msg = "لقد أدخلت التاريخ: " & UserInput
  Msg = Msg + vbCrLf + "ويوم التاريخ الذي أدخلته هو: " _
                    & DayOfWeek
  MsgBox Msg
End Sub
```

❑ احفظ المشروع باختيار البند **Save Project** من قائمة **File**.

❑ نفّذ برنامج مربعات الحوار وتمرن على عناصر القائمة الجديدة.

❑ اختر البند **إدخال نص** من قائمة **مربعات الحوار**.

يستجيب البرنامج بإظهار مربع الحوار المبين في الشكل 6-7.



الشكل 6-7

مربع حوار إدخال نص.

- اكتب أي نص في مربع الحوار هذا، ثم انقر على الزر موافق.

يستجيب البرنامج بإظهار رسالة تخبرك ماذا كتبت.

- اختر البند **إدخال نص**، ولكن هذه المرة انقر على الزر **إلغاء الأمر** بدلاً من الزر موافق.

*يستجيب البرنامج بإظهار رسالة تخبرك بأنك لم تكتب شيئاً، أو أنك نقرت الزر **إلغاء الأمر**.*

- اختر البند **إدخال رقم** من قائمة البرنامج.

يستجيب البرنامج بإظهار مربع حوار يطلب منك إدخال رقم.

- اكتب أي رقم صحيح مثل 1234، ثم انقر على الزر موافق.

يستجيب البرنامج بإظهار رسالة بالرقم الذي كتبته.

- اختر البند **إدخال رقم** مجدداً، ولكن لا تكتب رقماً هذه المرة مثل ABCD.

يستجيب البرنامج بإظهار رسالة تخبرك بأنك أدخلت رقماً غير صحيح.

- تمرن الآن على البند **إدخال تاريخ**، بإدخال صيغتي تاريخ إحداهما صحيحة والأخرى غير صحيحة في مربع الحوار **إدخال تاريخ**. التاريخ الصحيح يمكن أن يحمل صيغة مشابهة للصيغ التالية: July 4,1776 و January 1,1994 و 01/01/1994. أما الأمثلة على الصيغ الغير صحيحة فكثيرة منها: 1234,ABCD و 14/77/93.

*لاحظ أنه بعد إدخالك للتاريخ الصحيح في مربع الحوار **إدخال تاريخ** فإن البرنامج يُظهر رسالة تحمل التاريخ الذي أدخلته إضافة إلى اليوم من ذلك الأسبوع التابع لهذا التاريخ. فمثلاً يستجيب البرنامج عند إدخال التاريخ 2000/1/1 بإظهار الرسالة:*

لقد أدخلت التاريخ 2000/1/1 ويوم التاريخ الذي أدخلته هو: السبت.

نص الإجراء ()mnuGetString_click:

ينفذ الإجراء ()muuGetString_click عند اختيار البند **إدخال نص** من القائمة. ويبدأ الإجراء بتنفيذ التابع الوظيفي :()InputBox

```
UserInput = _
  InputBox("اكتب هنا أي نص:", "مثال على مربع الإدخال")
```

تُظهِر هذه العبارة، مربع الحوار المبين في الشكل 6-7، ويتم إسناد القيمة المعادة من التابع الوظيفي ()InputBox إلى المتحول UserInput، تساوي القيمة المعادة من التابع الوظيفي "لا شيء" null، إذا لم يكتب المستخدم أي شيء في مربع الإدخال. أما إذا كتبت أي شيء ونقرت على الزر **OK** فالقيمة المعادة من التابع الوظيفي سوف تساوي الشيء الذي كتبته.

تستخدم العبارة If التالية للتحقق من الضغط على الزر **Cancel**:

```
If UserInput = "" Then
  MsgBox "أنت لم تكتب أي شيء أو أنك ضغطت الزر إلغاء الأمر"
  Exit Sub
End If
```

يتحقق الشرط "" = UserInput عند عدم إدخال أي شيء أو عند ضغط الزر **Cancel**، ونتيجة ذلك تظهر الرسالة "أنت لم تكتب أي شيء أو أنك ضغطت الزر إلغاء الأمر". وينتهي الإجراء بواسطة العبارة Exit Sub. أما إذا لم يتحقق الشرط "" = UserInput، فسوف تنفذ آخر عبارة في الإجراء:

```
MsgBox "لقد كتبت ما يلي: " & UserInput
```

نص الإجراء ()mnuGetNumber_Click

يُنفذ الإجراء ()mnuGetNumber_Click عند اختيار البند **إدخال رقم** من قائمة مربعات الحوار، يطلب منك إدخال رقم ما، ثم يتحقق بأن الرقم صحيح.

يبدأ الإجراء بتنفيذ التابع الوظيفي ()InputBox:

```
UserInput = _
```

```
  InputBox("اكتب هنا أي رقم:" , "مثال على مربع الإدخال")
```

تُستخدم العبارة If للتحقق من النقر على الزر **Cancel:**

```
If UserInput = "" Then
  MsgBox "أنت لم تكتب أي شيء أو أنك ضغطت الزر إلغاء الأمر"
  Exit Sub
End If
```

يتحقق الشرط "" = UserInput عند نقر الزر **Cancel**، أو عند عدم إدخال أي شيء ونتيجة ذلك تظهر العبارة: "أنت لم تكتب أي شيء أو أنك ضغطت الزر إلغاء الأمر". وينتهي تنفيذ الإجراء بواسطة العبارة Exit Sub.

أما إذا أدخل المستخدم شيئاً، ولم يضغط على الزر **Cancel**، عندها يتأكد الإجراء بأن الرقم الذي أدخله المستخدم صحيح وذلك بواسطة التابع الوظيفي ()IsNumeric:

```
If Not IsNumeric(UserInput) Then
  MsgBox "الرقم الذي كتبته غير صحيح"
  Exit Sub
End If
```

فإذا كان المتحول UserInput قد ملئ برموز مثل ABC، يكون الشرط:

```
Not Is Numeric(UserInput)
```

محققاً ونتيجة لذلك ينفذ الجزء الواقع تحت عبارة الشرط If، فتظهر الرسالة "الرقم الذي كتبته غير صحيح". وينتهي الإجراء بواسطة العبارة Exit Sub. أما إذا كان المتحول UserInput مملوءاً برموز مثل 123، فعندها يصبح الشرط السابق غير محقق وتنفذ آخر عبارة في الإجراء أي:

```
MsgBox "لقد كتبت الرقم التالي: " + UserInput
```

بحيث تُظهر هذه العبارة الرقم الذي أدخلته.

الإجراء ()mnuGetdate_Click

ينفذ الإجراء ()mnuGetdate_Click عند اختيار البند **إدخال تاريخ** من قائمة مربعات الحوار. يطلب منك الإجراء إدخال تاريخ ثم يتحقق من أن التاريخ الذي أدخلته صحيح. يبدأ الإجراء بتنفيذ التابع الوظيفي ()InputBox:

```
UserInput = _
  InputBox("اكتب هنا أي تاريخ:" , "مثال على مربع الإدخال")
```

تستخدم عبارة If كما تعودنا للتحقق من النقر على الزر **Cancel** أو من عدم إدخال شيء:

```
If UserInput = "" Then
  MsgBox "أنت لم تكتب أي شيء أو أنك ضغطت الزر إلغاء الأمر"
  Exit Sub
End If
```

فعند عدم إدخال أي شيء أو عند نقر الزر **Cancel** يتحقق الشرط "" = UserInput، وتظهر الرسالة "أنت لم تكتب أي شيء أو أنك ضغطت الزر إلغاء الأمر".

يلي خطوة التحقق السابقة خطوة يتأكد منها الإجراء من إدخال التاريخ بصيغة صحيحة وذلك بواسطة التابع الوظيفي ()IsDate:

```
If Not IsDate(UserInput) Then
  MsgBox "التاريخ الذي كتبته غير صحيح"
  Exit Sub
End If
```

يعتبر الشرط Not IsDate(UserInput) محققاً عندما يكون المتحول UserInput مملوءاً برموز لا تلبي صيغة التاريخ الصحيحة أي قد يكون UserInput مملوءاً برموز مثل 123 أو أ ب ت.

ونتيجة ذلك، تظهر الرسالة "التاريخ الذي كتبته غير صحيح"، وينتهي الإجراء بواسطة العبارة Exit Sub. وعلى العكس من ذلك لا يتحقق الشرط إذا كانت صيغة التاريخ قد أدخلت بشكل صحيح.

ينتقل التنفيذ نتيجة ذلك إلى آخر أربع عبارات في الإجراء:

```
DayOfWeek = Format(UserInput, "dddd")
Msg = "لقد أدخلت التاريخ: " & UserInput
Msg = Msg + vbCrLf + "ويوم التاريخ الذي أدخلته هو: " _
                     & DayOfWeek
MsgBox Msg
```

تظهر هذه العبارات التاريخ الذي أدخله المستخدم بواسطة المتحول UserInput، إضافة إلى اسم اليوم التابع لذلك التاريخ عبر المتحول DayOfweek، يتم تحديث قيمة المتحول

DayOfweek بواسطة التابع ()Format. فمثلاً إذا كان المتحول UserInput يساوي 2000/1/1، فعندها يعود التابع Format(UserInput , "dddd") بالقيمة "السبت".

استخدام الوسائط الأخرى للتابع InputBox

استخدَم برنامج مربعات الحوار، التابع الوظيفي ()InputBox مع وسيطين فقط: الرسالة التي سيتم إظهارها وعنوان مربع الحوار. يمكنك أيضا استخدام وسائط اختيارية أخرى مع التابع المذكور، يحدد الوسيط الثالث قيمة افتراضية تظهر في الحقل النصي لمربع الحوار. أما الوسيطين الرابع والخامس فيحددان موقع مربع الحوار على الشاشة بوحدات مقدرة بواحدة Twips. فمثلاً استخدم العبارة التالية:

```
MyNumber=InputBox(":أدخل رقما" , "مثال", "7" , 100 ,200)
```

لإظهار مربع حوار يحمل الخصائص التالية:

الرسالة هي: أدخل رقماً.

العنوان: مثال.

القيمة الافتراضية التي ستظهر في الحقل النصي هي: 7.

تبعد الحافة اليسارية لمربع الحوار بمقدار 100 twips عن الحافة اليسارية للشاشة.

تبعد الحافة العلوية لمربع الحوار بمقدار 200 twips عن قمة الشاشة.

نتيجة ذلك، يظهر مربع الحوار عند احداثيي الشاشة X=100 و Y=200، كما أن القيمة الافتراضية الموجودة في الحقل النصي تساوي 7. يمكن للمستخدم النقر على **OK** في مربع الحوار، أو يمكنه إدخال رقمٍ آخر بدلاً من الرقم 7 ويتبعه بالنقر على **OK**.

مربعات الحوار المخصصة

يتم تفصيل مربعات الحوار المخصصة حسب ما يتضح من اسمها، من قِبل المستخدم (بينما كانت مربعات الحوار مسبقة التعريف، تسمح لك بوضع أشياء محددة مثل الأزرار والرموز).

تتشابه عملية تصميم مربع حوار مخصص من تصميم النموذج Form.

بل إن مربع الحوار المخصص بالواقع، هو نموذج عادي يستخدم لإظهار والحصول على المعلومات من المستخدم، وبمجرد الانتهاء من تصميم مربع الحوار المخصص، يصبح بالوسع استخدامه من قبل أي برامج أخرى مكتوبة بلغة فيجول بيسك.

سنعمل على تحسين برنامج مربعات الحوار بغية توضيح كيفية تصميم واستخدام مربع حوار مخصص في البرنامج. وسنصمم مربع حوار مخصص يدعى GetMonth.frm ويسمح للمستخدم اختيار شهر ما. وكعادتنا بعد الانتهاء من طور التصميم سنضيف إلى برنامج مربعات الحوار النص اللازم لاستخدام مربع الحوار GetMonth.frm.

تصميم مربع حوار مخصص

استخدم الخطوات التالية لتصميم مربع الحوار GetMonth.Frm:

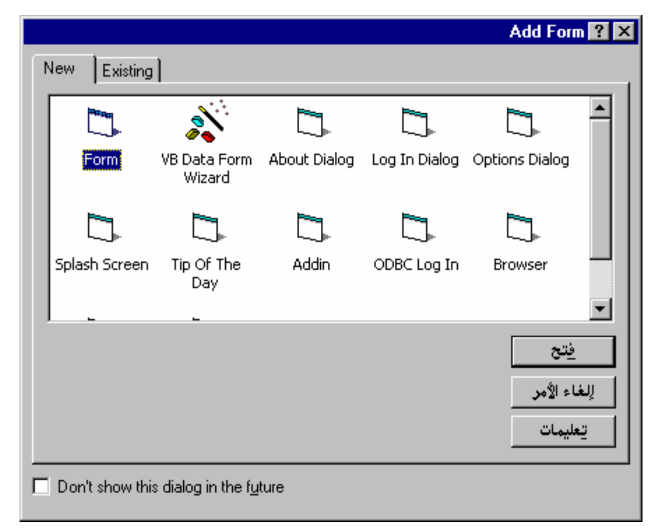
□تأكد أن المشروع Dialogs.VBp هو المشروع المفتوح حالياً في فيجول بيسك ثم اختر البند **Add From** من قائمة **Project**.

*يستجيب فيجول بيسك بإظهار الإطار **Add Form**. انظر الشكل 6-8.*

□اختر الرمز Form من الإطار **Add Form** ثم انقر على الزر **فتح**.

يستجيب فيجول بيسك بإضافة نموذج جديد يدعى Form1 إلى المشروع Dialogs.vbp.

□احفظ النموذج الجديد بالاسم GetMonth.Frm وذلك باختيار البند **Save Form1 As** من قائمة **File**، ثم استخدم مربع الحوار **Save File As** لحفظ النموذج بالاسم GetMonth.frm في الدليل C:\VB5Prg\Ch06.



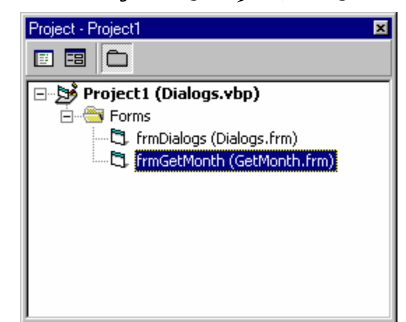
الشكل 6-8

الإطار **Add Form**.

قبل أن نذهب أبعد من ذلك، دعنا نتحقق بأن المشروع Dialogs.Vbp يحوي الآن نموذجين داخله.

□اختر البند **Project Explorer** من قائمة **View** لإظهار الإطار Project.

يظهر الآن الإطار Project بشكل الشاشة المبينة في الشكل 6-9.



الشكل 6-9

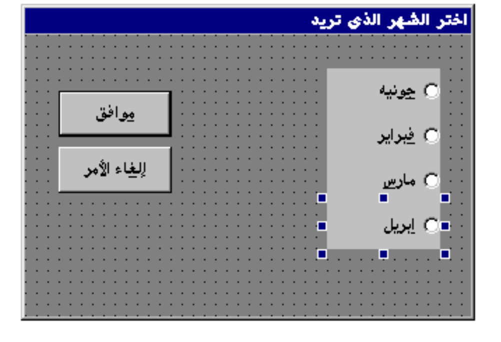
يُظهر الإطار Project بأن المشروع يمتلك نموذجين داخله:

Dialogs.Frm

و GetMonth.Frm

عليك الآن بناء مربع الحوار GetMonth.Frm طبقاً للجدول 6-8.

يفترض أن يظهر مربع الحوار المكتمل كما في الشكل 6-10.



الشكل 6-10

النموذج frmGetMonth

(طور التصميم).

▫احفظ العمل المنجز باختيار **Save Project** من القائمة **File** فإذا كنت قد أجريت تعديلات على النموذج frmDialogs أو على النموذج GetMonth.Frm فسوف تُحفظ هذه التعديلات.

الجدول 6–8. جدول خصائص مربع الحوار GetMonth.Frm.

الكائن	الخاصية	القيمة
Form	**Name**	**frmGetMonth**
	Caption	اختر الشهر الذي تريد
	BackColor	Dark gray
	BorderStyle	1-Fixed Single
	ControlBox	False
	MaxButton	False
	MinButton	False
	RightToLeft	True
CommandButton	**Name**	**cmdOK**
	Caption	&موافق
	Cancel	False
	Default	True
	RightToLeft	True
CommandButton	**Name**	**cmdCancel**
	Caption	إل&غاء الأمر
	Cancel	True
	Default	False

	RightToLeft	True
Option Button	**Name**	**optJan**
	Caption	&جونيه
Option Button	**Name**	**optFeb**
	Caption	&فبراير
Option Button	**Name**	**optMar**
	Caption	&مارس
Option Button	**Name**	**optApr**
	Caption	&ابريل

ملاحظة

يطالبك الجدول 6–8 بوضع أربعة أزرار خيارات Option Buttons لأربع شهور فقط، ولهذا فالمستخدم سيكون قادراً على اختيار أحد الشهور التالية:
جونيه، فبراير، مارس، إبريل. والسبب وراء ذلك رغبتنا باختصار زمني التصميم وكتابة البرنامج. لكنك تستطيع تحسين نموذج frmGetMonth.frm لشمل الشهور الاثني عشر كاملة.

خصائص مناسبة لمربعات الحوار

يصمم مربع الحوار النموذجي، بحيث لا يمكن تغيير شكل ظهوره أثناء زمن التنفيذ. وكما يتبين لنا من الجدول 6–8 فإن خصائص النموذج frmGetMonth قد أُعدّت بحيث لا تسمح لك بتكبير أو تصغير مربع الحوار أثناء زمن التنفيذ.

وُضعت الخاصية BorderStyle لمربع الحوارfrmGetMonth على الحالة Fixed Single لكي لا يسمح بتغيير حجم مربع الحوار خلال زمن التنفيذ.

وضعت الخاصية ControlBox لمربع الحوار frmGetMonth على الحالة False مما يعني أن مربع الحوار سيظهر بدون مربع قائمة التحكم Control Menu Box (وهو ذلك المربع الذي يظهر في الزاوية اليسرى العليا من النموذج).

وضعت الخاصية MaxButton لمربع الحوار frmGetMonth على الحالة False

مما يعني أن مربع الحوار سيظهر بدون زر التكبير في زاويته العليا اليمنى. مما يمنع المستخدم من تكبير مربع الحوار أثناء زمن التنفيذ.

وضعت الخاصية MinButton لمربع الحوار على الحالة False مما يعني أن مربع الحوار سيظهر بدون زر التصغير في الزاوية اليمنى العليا. مما يمنع المستخدم من تصغير مربع الحوار أثناء زمن التنفيذ.

الخاصيتان Cancel و Default لأزرار الأوامر

كما تبين لنا من الجدول 6-8 فقد وضعت الخاصية Default للزر **موافق** على الحالة True كما وضعت الخاصية Cancel للزر **إلغاء الأمر** على الحالة True أيضاً.

يؤدي تعيين حالة الخاصية Default لزر الأمر على القيمة True إلى جعل زر الأمر هذا وكأنه زر افتراضي Default وهو ذلك الزر الذي يُعتبر أنه قد نقر عليه، إذا تم الضغط على المفتاح Enter.

ولذلك وبسبب وضع الخاصية Default للزر **موافق** على الحالة True فإن الضغط على المفتاح Enter عندما يكون مربع الحوار frmGetMonth ظاهراً أثناء زمن التنفيذ، يتسبب بنفس الأثر الذي ينجم عن نقر الزر **موافق**.

يؤدي تعيين حالة الخاصية Cancel لزر الأمر على القيمة True إلى جعل هذا الزر بمثابة مفتاح الهروب Escape، (هو الزر الذي يتم اختياره نتيجة الضغط على المفتاح Esc الموجود في لوحة المفاتيح). لذلك ونتيجةً لوضع الخاصية Cancel للزر **إلغاء الأمر** على الحالة True، فإن الضغط على المفتاح Esc أثناء ظهور مربع الحوار خلال زمن التنفيذ، يتسبب بنفس الأثر الذي ينجم عن نقر الزر **إلغاء الأمر**.

إظهار و إخفاء مربع حوار محدد

□استخدم الطريقة Show لإظهار مربع حوار مخصص. المفروض في مربع الحوار أن يظهر كمربع حوار محدد Modal، (بمعنى أن المستخدم يجب أن يستجيب له، قبل أن يتمكن من استئناف تنفيذ البرنامج)، ولإظهار مربع الحوار

كمربع حوار محدد، تستخدم الطريقة Show مع متحول وسيطي يساوي الواحد. فمثلاً لإظهار frmMyDialog كمربع حوار محدد استخدم العبارة التالية:

```
FrmMyDialog.Show 1
```

❑استخدم الطريقة Hide لإخفاء مربع حوار. فمثلاً استخدم العبارة اللاحقة لإخفاء مربع الحوار frmMyDialog الظاهر حالياً:

```
FrmMyDialog.Hide
```

سنضيف نص برنامج إلى برنامج مربعات الحوار، لتباين كيفية عمل الطريقتين Show و Hide على أرض الواقع.

سيُظهر نص البرنامج هذا، مربع الحوار frmGetMonth عندما ينتقي المستخدم عنصر القائمة المدعو **الحصول على شهر**، ثم يخفي ذلك المربع عندما يستجيب المستخدم له بالنقر على أحد الزرين: **موافق** أو **إلغاء الأمر** في ذلك المربع.

❑أضف عنصرين جديدين مع المواصفات التالية إلى قائمة النموذج frmDialogs (ضعها فوق البند **خروج**).

...الحصول على الشهر (مربع حوار مخصص)	mnuGetMonth
...–	mnuSep3

و بعد إضافة هذين العنصرين يفترض أن يبدو جدول قائمة البرنامج Dialogs كذاك المبين في الجدول 6-9.

ملاحظة

استخدم الخطوتين التاليتين لإضافة عناصر القائمة إلى النموذج frmDialogs:

– اختر النموذج frmDialogs.

– اختر **Menu Editor** من القائمة **Tools**.

يستجيب فيجول بيسك بإظهار Menu Editor وإظهار قائمة النموذج frmDialogs.

إذا كان النموذج frmGetMonth هو النموذج الحالي أي الإضاءة متوضعة عليه، ولم تختر النموذج frmDialogs قبل اختيار **Menu Editor** من القائمة **Tools**، فسوف يعتقد فيجول بيسك بأنك ترغب باستخدام محرر القوائم لتعديل قائمة النموذج

.frmGetMonth

الجدول 6-9. قائمة برنامج مربعات الحوار بعد إضافة البند الحصول على شهر.

الخاصية Name	العنوان
mnuDialogs	&مربعات الحوار
mnuOkCancel	...مربع الحوار موافق – إلغاء الأمر
mnuAbortRetryIgnore	...مربع الحوار إحباط – إعادة المحاولة– تجاهل
mnuYesNoCancel	...مربع الحوار نعم – لا – إلغاء الأمر
mnuYesNo	...مربع الحوار نعم – لا
mnuRetryCancel	...مربع الحوار إعادة المحاولة – إلغاء الأمر
mnuSep1	...–
mnuGetString	...إدخال نص
mnuGetNumber	...إدخال رقم
mnuGetDate	...إدخال تاريخ
mnuSep2	...–
mnuGetMonth	...الحصول على الشهر (مربع حوار مخصص)
mnuSep3	...–
mnuExit	...&خروج

نص الإجراء ()mnuGetMonth_Click

```
Private Sub mnuGetMonth_Click()
```

```
'frmGetMonth إظهار مربع الحوار المخصص
frmGetMonth.Show vbModal

'إظهار الشهر الذي اختاره المستخدم
'إذا قام بعملية الاختيار
If frmGetMonth.Tag = "" Then
  MsgBox "لقد ألغيت عملية الإختيار"
Else
  MsgBox "لقد اخترت الشهر : " & frmGetMonth.Tag
End If
End Sub
```

يُنفذ الإجراء mnuGetMonth_Click() عند اختيار البند **الحصول على شهر** من قائمة **مربعات الحوار**، يستخدم نص برنامج هذا الإجراء الطريقة Show لإظهار مربع الحوار frmGetMonth كما يظهر الشهر الذي اختاره المستخدم، بإظهار الخاصية Tag لمربع الحوار frmGetMonth، وستلاحظ قريباً أن نص برنامج مربع الحوار frmGetMonth استخدم الخاصية Tag للنموذج frmGetMonth (أي frmGetMonth.Tag) لحفظ اسم الشهر الذي اختاره المستخدم.

- تحقق أن العبارة Option Expliert تظهر في قسم التصاريح العامة للنموذج frmGetMonth بالشكل التالي:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

ملاحظة

لإظهار إطار برنامج النموذج frmGetMonth انقر نقرة مزدوجة بواسطة الفأرة فوق أي مكان من النموذج، ليظهر إطار برنامج النموذج.

تأكدنا للتو من وجود العبارة Option Explicit في قسم التصاريح العامة للنموذج frmDialogs. يعني هذا أنه يتوجب عليك التصريح عن أي متحول قبل استخدامه في إجراءات النموذج frmDialogs.

وبأسلوب مشابه يجب أن تكون العبارة Option Explicit موجودة في قسم التصاريح العامة للنموذج frmGetMonth. مما يعني أنه يتوجب التصريح عن أي متحول قبل استخدامه في إجراءات النموذج frmGetMonth.
لهذا لا بد من استخدام العبارة Option Explicit مع كل نموذج من النماذج.

نص الإجراء cmdOK_Click()

```
Private Sub cmdOk_Click()

    'frmGetMonth.Tag تغيير قيمة الخاصية
    'إلى الشهر الذي اختاره المستخدم
    'Form تستخدم هذه الخاصية كخرج لهذا النموذج
    If optJan.Value = True Then frmGetMonth.Tag = "جونيه"
    If optFeb.Value = True Then frmGetMonth.Tag = "فبراير"
    If optMar.Value = True Then frmGetMonth.Tag = "مارس"
    If optApr.Value = True Then frmGetMonth.Tag = "ابريل"

    'frmGetMonth إخفاء النموذج
    frmGetMonth.Hide

End Sub
```

ينفذ الإجراء cmdOK_Click() عند نقر الزر **موافق** في مربع الحوار frmGetMonth يستخدم نص برنامج هذا الإجراء سلسة من أربع عبارات للتعرف على زر الخيار المنتقى حالياً، ويعدل وفقاً لذلك، الخاصية Tag للنموذج frmGetMonth مكلفاً إياها بالشهر الذي تم انتقاؤه. يستخدم الإجراء بعد عبارات If الأربعة، طريقة Hide، لإخفاء مربع الحوار frmGetMonth من الشاشة.

تستخدم الخاصية Tag لمربع الحوار frmGetMonth كخرج لمربع الحوار. أي بمعنى أن باستطاعة من يعمل على مربع الحوار المذكور استخدام frmGetMonth.Tag لتحديد الشهر الذي تم اختياره.

إليك الآن عبارة If ذات الشكل:

```
    If optJan.Value = True Then frmGetMonth.Tag = "جونيه"
```

تعتبر العبارة هذه مطابقة للعبارة التالية:

```
If optJan.Value = True Then
   frmGetMonth.Tag = "جونيه"
End If
```

أي لا تعود هناك حاجة لعبارة End If، إذا اتسعت عبارة If على سطر واحد. يفيد هذا الأمر بتقليل الجهد البرمجي، بل و يسهل عملية قراءة وضع البرنامج الذي يحمل مجموعة من عبارات If المتلاحقة.

نص الإجراء cmdCancel_Click()

```
Private Sub cmdCancel_Click()

    'frmGetMonth تغيير قيمة الخاصية
    'إلى لا شيء
    frmGetMonth.Tag = ""

    'frmGetMonth إخفاء النموذج
    frmGetMonth.Hide

End Sub
```

ينفذ الإجراء cmdCancel_Click() عند نقر الزر **إلغاء الأمر**، في النموذج frmGetMonth. ويسند نص برنامج الإجراء هذا القيمة "" (أي Null) **إلى** الخاصية Tag للنموذج (أي frmGetMonth.Tag). ثم يستخدم الطريقة Hide لإخفاء مربع الحوار frmGetMonth من الشاشة.يعني تكليف frmGetMonth.Tag بالقيمة""، أن المستخدم نقر الزر **إلغاء الأمر**.

- احفظ العمل المنجز باختيار **Save Project** من القائمة **File** لفيجول بيسك.
- نفّذ برنامج مربعات الحوار، واختر **الحصول على شهر** من القائمة **مربعات الحوار**، وتمرّن على مربع الحوار المخصص.

لاحظ بشكل خاص، أن نقر أحد أزرار الخيارات، يؤدي إلى تمركز التحكم عنده. من وجهة نظر لوحة المفاتيح، نستطيع ضغط المفتاح Esc على لوحة المفاتيح، الأمر الذي يملك نفس التأثير الناتج عن نقر الزر **إلغاء الأمر** بواسطة الفأرة، كما أن ضغط

المفتاح Enter عندما يكون التحكم متمركزاً عند أزرار الخيار، يكافئ نقر الزر **موافق**، والسبب كما ذكرنا قبلاً، أن الخاصية Default للزر **موافق** تساوي True، كما أن الخاصية Cancel للزر **إلغاء الأمر** تساوي True أيضاً.

- اختر البند **خروج** من القائمة **مربعات الحوار** لإنهاء البرنامج.

مربعات الحوار الشائعة

يتضمن فيجول بيسك عنصر تحكم، يُطلق عليه الاسم CommonDialog (أي مربعات الحوار الشائعة)، يمكن استخدام عنصر التحكم هذا، لإظهار مربعات الحوار الشائعة أثناء زمن التنفيذ بمجرد تعيين خصائصها. يُستخدم مربع الحوار الشائع غالباً، كمربع حوار يسمح للمستخدم باختيار وحفظ الملفات.

برنامج مربعات الحوار الشائعة

سنكتب الآن برنامج مربعات الحوار الشائعة، لتوضيح كيفية استخدام عنصر التحكم CommonDialog في برنامج ما. يُظهر هذا البرنامج، مربعي حوار: مربع الحوار اختيار لون ومربع الحوار فتح.

التمثيل المرئي لبرنامج مربعات الحوار الشائعة

سنبدأ كعادتنا بتمثيل نموذج البرنامج:

- ابدأ مشروعاً من النوع التنفيذي القياسي Standard EXE.
- احفظ نموذج المشروع بالاسم Common.Frm في الدليل C:\VB5Prg\Ch06.
- احفظ ملف المشروع بالاسم Common.Vbp في الدليل نفسه.

يعتبر زر الأمر Command Button مثالاً على عنصر تحكم، يظهر دائماً في إطار مربع الأدوات، كما أن مربع النص وزر الخيار كلها أمثلة على عناصر تحكم تظهر دائماً في إطار مربع الأدوات.

من جهة أخرى، يُعتبر عنصر التحكم Common Dialog مثالاً على عنصر تحكم لا يظهر دائماً في مربع الأدوات، و إنما يضاف أو يزال من قبل المستخدم.

يطلق على عنصر التحكم مربعات الحوار الشائعة، اسم عنصر التحكم OCX. وكما ذكرنا في الفصل الأول، تحت عنوان كتابة أول برنامج لك في فيجول بيسك، أنه يتوجب على المستخدم التحقق من وجود الملف Msvbvm50.DLL في الدليل System. وهو ملف يحوي المكتبات الضرورية لتشغيل برنامجك التنفيذي. فمثلاً، يمكّن البرنامج من استخدام القوائم والفأرة وكل المظاهر الأخرى التي قد يحتويها ملفك التنفيذي. ومن بين الأشياء التي يحويها الملف Msvbvm50.DLL هي عناصر التحكم التي تشاهد دوماً في إطار مربع الأدوات، وهو السبب الذي يمكّنك من توزيع برامجك المكتوبة بلغة فيجول بيسك، دون توزيع ملفات إضافية ترافق عناصر التحكم هذه.

أما عناصر التحكم من أمثال مربعات الحوار الشائعة، فليست محتواة في الملف Msvbvm50.DLL، مما يعني أنه يتوجب عليك عند توزيع ملفاتك التنفيذية EXE إرفاقها بالملفات المرافقة لعناصر OCX التي يستخدمها برنامجك.

يدعى الملف الذي يرافق عنصر تحكم مربعات الحوار الشائعة، بالاسم Comdlg32.ocx ويفترض أن تشاهده ضمن الدليل System. (يتم تنصيب هذا الملف أثناء عملية إعداد فيجول بيسك).

لننظر الآن إذا كان عنصر التحكم CommonDialog محتوىً في المشروع Common.Vbp.

- اختر البند **Custom Components** من القائمة **Project** لفيجول بيسك.

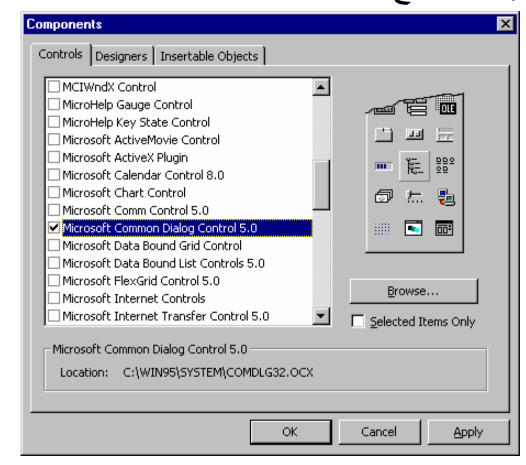
يستجيب فيجول بيسك بإظهار مربع الحوار Components (انظر الشكل 6-11).

يمتلك الإطار Components ثلاث صفحات في الأعلى، كما يظهر من الشكل 6-11.

- تحقق أن الصفحة Controls هي الظاهرة (كما في الشكل 6-11).
- تدرّج عبر البنود، حتى تصل للبند Microsoft CommonDialog Control 5.0، (انظر الشكل 6-11). إذا كانت علامة الاختيار ظاهرة في الخانة المجاورة لهذا البند، فهذا يعني أنه موجود مسبقاً في إطار مربع الأدوات.
- أما إذا لم تكن علامة الاختيار موجودة في الخانة المجاورة لهذا البند Microsoft CommonDialog Control 5.0، فانقر فوق الخانة لوضع علامة اختيار.
- انقر الزر **OK** في الإطار Component.

يضاف نتيجة ذلك، عنصر التحكم CommonDialog إلى مربع الأدوات.

ستضطر غالباً إلى سحب الحافة السفلية لمربع الأدوات Tool Box لتتمكن من مشاهدة عنصر التحكم CommonDialog، يوضح الشكل 6-12 عنصر التحكم CommonDialog في إطار مربع الأدوات.



الشكل 6-11

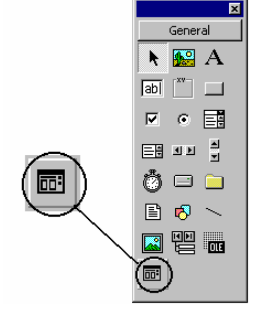
مربع الحوار

Components.

يظهر مستطيل أصفر عند وضع مؤشر الفأرة فوق عنصر التحكم هذا، لتظهر الرسالة CommonDialog داخل هذا المستطيل. وبهذه الطريقة تتحقق من وضع رمز عنصر التحكم CommonDialog في إطار مربع الأدوات.

- اختر النموذج Common.Frm بالنقر في أي مكان عليه.
- انقر نقرة مزدوجة على عنصر التحكم CommonDialog.

يستجيب فيجول بيسك بوضع عنصر التحكم CommonDialog في النموذج Common.Frm، والاسم الافتراضي الذي يطلقه فيجول بيسك على عنصر التحكم هذا هو CommonDialog1.



الشكل 6-12

رمز عنصر تحكم

مربعات الحوار الشائعة

في نافذة الأدوات.

- أنشئ النموذج Common.Frm طبقاً للجدولين 6-10 و 6-11 (الجدول 6-11 هذا

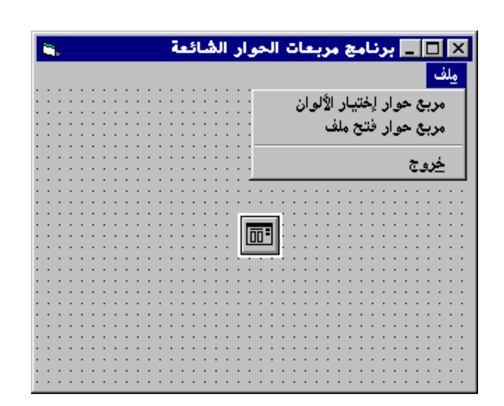
هو جدول قائمة النموذج frmCommon.

الجدول 6-10. جدول خصائص النموذج Common.Frm.

الكائن	الخاصية	القيمة
Form	**Name**	**frmCommon**
	Caption	برنامج مربعات الحوار الشائعة
	RightToLeft	True
Common Dialog	**Name**	**CommonDialog1**
	CancelError	True
Menu	(انظر الجدول 6-11)	(انظر الجدول 6-11)

الجدول 6-11. جدول قائمة النموذج frmCommon.

الاسم	العنوان
mnuFile	&ملف
mnuColor	...مربع حوار ا&ختيار الألوان
mnuOpen	...مربع حوار &فتح ملف
mnuSep1	-...
mnuExit	...&خروج



الشكل 6- 13

النموذج frmCommon.

كتابة نص برنامج مربعات الحوار الشائعة

سنبدأ الآن بإدخال نص برنامج مربعات الحوار الشائعة:

- تحقق من ظهور العبارة Option Explicit في قسم التصاريح العامة للنموذج frmCommon أي:

```
' يجب التصريح عن كل المتحولات
Option Explicit
```

- اكتب ما يلي في الإجراء ()mnuColor_Click:

```
Private Sub mnuColor_Click()
    'تشغيل متصيد الأخطاء لمعرفة أن المستخدم ضغط
    'زر إلغاء الأمر في مربع حوار الألوان
    On Error GoTo ColorError
    'إظهار مربع حوار الألوان
    CommonDialog1.ShowColor

    'تغيير لون خلفية البرنامج إلى اللون الذي
    'اختاره المستخدم
    frmCommon.BackColor = CommonDialog1.Color

    'الخروج من الإجراء
    Exit Sub

    ColorError:
    'ينفذ هذا الجزء من الإجراء عندما يضغط المستخدم
    'على زر إلغاء الأمر
    MsgBox "لقد ألغيت مربع حوار الألوان"
    Exit Sub

End Sub
```

- اكتب ما يلي في الإجراء ()mnuOpen_Click:

```
Private Sub mnuOpen_Click()

    Dim Filter As String

    'تشغيل متصيد الأخطاء لمعرفة أن المستخدم ضغط
    'زر إلغاء الأمر في مربع حوار فتح ملف
```

```
    On Error GoTo OpenError

    'تحديد أنواع الملفات المراد فتحها
    Filter = "كل الملفات (*.*)|*.*|"
    Filter = Filter + "الملفات النصية(*.txt)|*.txt|"
    Filter = Filter + "الملفات الدفعية(*.bat)|*.bat"

    CommonDialog1.Filter = Filter

    'تحديد نوع الملفات الافتراضي وهو الملفات النصية
    CommonDialog1.FilterIndex = 2

    'إظهار صندوق الحوار
    CommonDialog1.ShowOpen

    'إظهار اسم الملف الذي اختاره المستخدم
    MsgBox "لقد اخترت الملف: " & CommonDialog1.filename

    Exit Sub

    OpenError:
    'ينفذ هذا الجزء من الإجراء عندما يضغط المستخدم
    'على زر إلغاء الأمر
    MsgBox "لقد ألغيت مربع حوار فتح ملف"
    Exit Sub
End Sub
```

- اكتب ما يلي في الإجراء mnuExit_Click():

```
Private Sub mnuExit_Click()
    End
End Sub
```

- احفظ المشروع كاملاً.

تنفيذ برنامج مربعات الحوار الشائعة

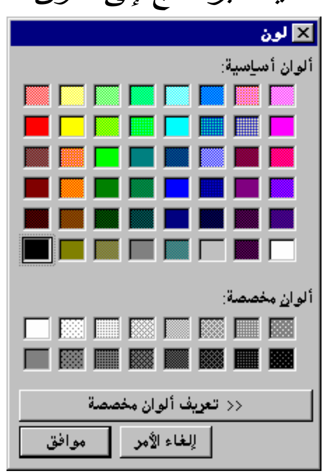
نفّذ برنامج مربعات الحوار الشائعة، ولاحظ أن عنصر التحكم CommonDialog لا يكون مرئياً أثناء مرحلة التنفيذ، وإنما يظهر حالما يتم اختيار **مربع حوار اختيار الألوان** أو **مربع حوار فتح ملف** من القائمة **ملف**.

- اختر **مربع حوار اختيار الألوان** من القائمة **ملف**.

يستجيب برنامج مربعات الحوار الشائعة بإظهار مربع الحوار لون، انظر الشكل 6-14.

- انتق لوناً من مربع حوار **لون**، بنقر اللون المطلوب، ثم انقر الزر **موافق**.

يستجيب برنامج مربعات الحوار الشائعة، بإغلاق مربع الحوار لون، وتبديل لون خلفية البرنامج إلى اللون المنتقى.



الشكل 6-14

مربع الحوار **لون**.

- اختر **مربع الحوار فتح ملف** من القائمة **ملف**.

*يستجيب البرنامج بإظهار مربع الحوار **فتح**، **انظر** الشكل 6-15.*

- اختر دليلاً محدداً على قرصك الصلب، واختر من مربع الملفات، البند **كل الملفات**، ثم اكتب *.* في مربع اسم الملف، واختر ملفاً ثم انقر الزر **فتح** في مربع الحوار فتح (انظر الشكل 6-15).

يستجيب برنامج مربعات الحوار الشائعة، بإظهار رسالة تخبرك اسم الملف الذي اخترته.



الشكل 6-15

مربع الحوار **فتح**.

□تمرن على برنامج مربعات الحوار الشائعة ثم اختر **خروج** من القائمة **ملف** لإنهاء عمل البرنامج.

كيف يعمل برنامج مربعات الحوار الشائعة

يستخدم البرنامج عنصر التحكم المخصص CommonDialog، لإظهار مربع الحوار **لون** ومربع الحوار **فتح**، يمكّنك مربع حوار اللون من اختيار لون ما من لوح الألوان، ويمكّنك مربع الحوار **فتح** من اختيار ملف.

مربع الحوار لون

يُنفذ الإجراء ()mnuColor_Click آلياً، عند اختيار **مربع حوار اختيار الألوان** من القائمة **ملف**، يستخدم نص برنامج هذا الإجراء، عنصر التحكم المخصص CommonDialog لإظهار مربع الحوار **لون**:

```
Private Sub mnuColor_Click()

    'تشغيل متصيد الأخطاء لمعرفة أن المستخدم ضغط
    'زر إلغاء الأمر في مربع حوار الألوان
    On Error GoTo ColorError

    'إظهار مربع حوار الألوان
    CommonDialog1.ShowColor

    'تغيير لون خلفية البرنامج إلى اللون الذي
    'اختاره المستخدم
```

```
    frmCommon.BackColor = CommonDialog1.Color

    'الخروج من الإجراء
    Exit Sub

    ColorError:
    'ينفذ هذا الجزء من الإجراء عندما يضغط المستخدم
    'على زر إلغاء الأمر
    MsgBox "لقد ألغيت مربع حوار الألوان"
    Exit Sub

End Sub
```

يضع الإجراء mnuColor_Click() مصيدة للأخطاء، قبل إظهاره لمربع الحوار:

```
On Error GoTo ColorError
```

وظيفة هذه المصيدة، مراقبة (أو تصيد) الخطأ أثناء إظهاره مربع الحوار. لنفترض جدلاً، أنك وضعت الخاصية CancelError لعنصر التحكم المخصص CommanDialog1 أثناء طور التصميم على الحالة True، وهكذا، سيؤدي ضغط الزر **إلغاء الأمر**، أثناء ظهور مربع الحوار إلى توليد خطأ. ونتيجة لمرحلة تصيد الخطأ السابقة، سيتحول تنفيذ البرنامج إلى اللافتة ColorError.

يُظهر الإجراء بعد مرحلة تصيد الخطأ، مربع الحوار **لون** عن طريق العبارة التالية:

```
CommonDialog1.ShowColor
```

كما اتفقنا، يؤدي نقر الزر **إلغاء الأمر** أثناء ظهور مربع الحوار إلى توليد خطأ، ونتيجة لذلك ينفذ جزء البرنامج الواقع تحت اللافتة ColorError. يُظهر جزء البرنامج هذا رسالة، ثم ينهي تنفيذ الإجراء:

```
ColorError:
'ينفذ هذا الجزء من الإجراء عندما يضغط المستخدم
'على زر إلغاء الأمر
MsgBox "لقد ألغيت مربع حوار الألوان"
Exit Sub
```

إما إذا لم تضغط الزر **إلغاء الأمر**، وإنما اخترت بدلاً من ذلك لوناً من لوح الألوان، ثم أتبعت ذلك بنقر الزر **موافق**. فسينفذ الإجراء عند ذلك، العبارتين التاليتين:

```
frmCommon.BackColor = CommonDialog1.Color
```

```
Exit Sub
```

تُبدّل العبارة الأولى لون النموذج إلى اللون الذي اخترته، والذي يتحدد بالخاصية Color لعنصر التحكم CommonDialog1 (أي CommonDialog1.Color). وتنهي العبارة الثانية تنفيذ الإجراء.

مربع الحوار فتح

ينفذ الإجراء mnuOpen_Click() عند اختيار **فتح ملف** من القائمة **ملف**. يستخدم نص هذا البرنامج عنصر التحكم CommonDialog1 لإظهار مربع الحوار **فتح**:

```
Private Sub mnuOpen_Click()

    Dim Filter As String

    'تشغيل متصيد الأخطاء لمعرفة أن المستخدم ضغط
    'زر إلغاء الأمر في مربع حوار فتح ملف
    On Error GoTo OpenError

    'تحديد أنواع الملفات المراد فتحها
    Filter = "كل الملفات (*.*)|*.*|"
    Filter = Filter + "الملفات النصية(*.txt)|*.txt|"
    Filter = Filter + "الملفات الدفعية(*.bat)|*.bat"

    CommonDialog1.Filter = Filter

    'تحديد نوع الملفات الافتراضي وهو الملفات النصية
    CommonDialog1.FilterIndex = 2

    'إظهار صندوق الحوار
    CommonDialog1.ShowOpen

    'إظهار اسم الملف الذي اختاره المستخدم
    MsgBox "لقد اخترت الملف: " & CommonDialog1.filename

    Exit Sub

    OpenError:
```

```
    'ينفذ هذا الجزء من الإجراء عندما يضغط المستخدم
    'على زر إلغاء الأمر
    MsgBox "لقد ألغيت مربع حوار فتح ملف"
    Exit Sub
End Sub
```

تعد أول عبارة في الإجراء فخاً للخطأ الممكن وقوعه:

```
    On Error GoTo ColorError
```

وبنتيجة مرحلة التصيد هذه، يتسبب نقر الزر **إلغاء الأمر** أثناء ظهور مربع الحوار فتح إلى تولد خطأ أثناء مرحلة التنفيذ، مما يؤدي إلى انتقال التنفيذ إلى اللافتة :OpenError. يستخدم الإجراء بعد مرحلة التصيد هذه، الخاصية Filter لعنصر التحكم CommonDialog1 لملء بنود مربع سرد أنواع الملفات الموجود في مربع **فتح**.

يُملأ مربع سرد أنواع الملفات بثلاثة أنواع:

```
    Filter = "كل الملفات (*.*)|*.*|"
    Filter = Filter + "الملفات النصية(*.txt)|*.txt|"
    Filter = Filter + "الملفات الدفعية(*.bat)|*.bat"
    CommonDialog1.Filter = Filter
```

تَملأ العبارات السابقة، الخاصية Filter لعنصر التحكم CommonDialog1 بسلسة تُحدد نصوص البنود التي ستظهر في مربع سرد الملفات، إضافة إلى أنواع هذه الملفات: (*.* و *.txt و *.bat).

فمثلاً يتحدد أول بند يظهر في مربع سرد الملفات بالسلسة التالية:

```
    Filter = "كل الملفات (*.*)|*.*|"
```

السلسة مؤلفة من جزئين، ينفصلان بالحرف (|)، ويمكن الحصول عليه بالضغط على Shift+\. يُشير الجزء الأول من السلسلة إلى النص الذي يظهر في مربع سرد الملفات:

```
    كل الملفات (*.*)
```

أما الجزء الثاني، فيشير إلى هيكلية الملفات التي ستظهر عند اختيار هذا البند أي عند اختيار:

```
     *.*
```

بمعنى أن اختيار البند الأول من مربع سرد الملفات، سيؤدي إلى ظهور الملفات ذات الهيكلية *.* (كل الملفات) في مربع الحوار.

لاحظ أن العبارات السابقة عمدت إلى ملء الخاصية Filter لعنصر التحكم CommDialog1 على مراحل، وذلك بهدف التوضيح. فتم أولاً ملء المتحول Fitter:

```
Filter = "كل الملفات (*.*)|*.*|"
Filter = Filter + "الملفات النصية(*.txt)|*.txt|"
Filter = Filter + "الملفات الدفعية(*.bat)|*.bat"
```

ثم كلفت الخاصية Filter لعنصر التحكم CommonDialog1 بقيمة المتحول Filter:

```
CommonDialog1.Filter = Filter
```

بعد الانتهاء من ملء مربع سرد الملفات الموجود في مربع الحوار **فتح**، يتم تحديد البند الافتراضي الذي سيظهر في مربع سرد الملفات، وهو هنا البند 2، وتتجز هذه الخطوات بإسناد القيمة 2 إلى الخاصية FilterIndex للعنصر CommonDialog1.

```
CommonDialog1.FilterIndex = 2
```

وباعتبار البند الثاني في مربع سرد الملفات، هو الملفات النصية (*.txt) فالملفات الافتراضية التي ستندرج في مربع الحوار **فتح**، هي الملفات ذات الامتداد TXT.

يُظهر الإجراء، مربع الحوار **فتح** بواسطة العبارة التالية:

```
CommonDialog1.ShowOpen
```

يتولد خطأ في مرحلة التنفيذ، إذا تم نقر الزر **إلغاء الأمر** أثناء ظهور مربع الحوار، ونتيجة لذلك، يُنفذ جزء البرنامج الواقع تحت اللافتة OpenError ، يُظهر هذا الجزء رسالة، وينهي تنفيذ البرنامج OpenError:

```
OpenError:
'ينفذ هذا الجزء من الإجراء عندما يضغط المستخدم
'على زر إلغاء الأمر
MsgBox "لقد ألغيت مربع حوار فتح ملف"
Exit Sub
```

أما إذا لم يتم النقر على الزر **إلغاء الأمر**، وإنما اختار المستخدم ملفاً من مربع الحوار، فسوف يتم إظهار اسم الملف وينتهي الإجراء:

```
MsgBox "لقد اخترت الملف: " & CommonDialog1.filename
Exit Sub
```

يقع اسم الملف الذي اخترته، في الخاصية CommonDialog1.Filename.

مربعات الحوار الشائعة الأخرى

يستخدم برنامج مربعات الحوار الشائعة، عنصر التحكم CommanDialog، لإظهار مربعي الحوار **لون** و **فتح**. يمكن في الواقع استخدام عنصر التحكم هذا، لإظهار مربعات الحوار الشائعة الأخرى، يُدرج الجدول 6–12 مربعات الحوار الشائعة التي يمكن إظهارها، والقيم التي يجب إسنادها للخاصية Action حتى تظهر هذه المربعات. فمثلاً لإظهار مربع الحوار **حفظ باسم**، يتوجب عليك إسناد القيمة 2 إلى الخاصية Action:

```
CommonDialog1.Action = 2
```

أو بواسطة العبارة التالية الأكثر وضوحاً:

```
CommonDialog1.ShowSave
```

الجدول 6–12. مربعات الحوار الشائعة، وكيفية إظهارها.

لإظهار مربع الحوار	أسند للخاصية Action القيمة	الطريقة المباشرة
فتح	1	ShowOpen
حفظ باسم	2	ShowSave
الألوان	3	ShowColor
الخطوط	4	ShowFont
الطابعة	5	ShowPrinter

وكما كان عليه الحال مع مربعي الحوار **فتح** و **لون**، تستطيع استخدام خصائص عنصر التحكم CommonDialog، لتحديد استجابة المستخدم لمربع الحوار.

نماذج ومربعات حوار أخرى

تعلمت حتى الآن، كيفية إنشاء مربعات حوار باستخدام العبارة MsgBox، والتابع الوظيفي ()MsgBox، وباستخدام عنصر التحكم CommonDialog.

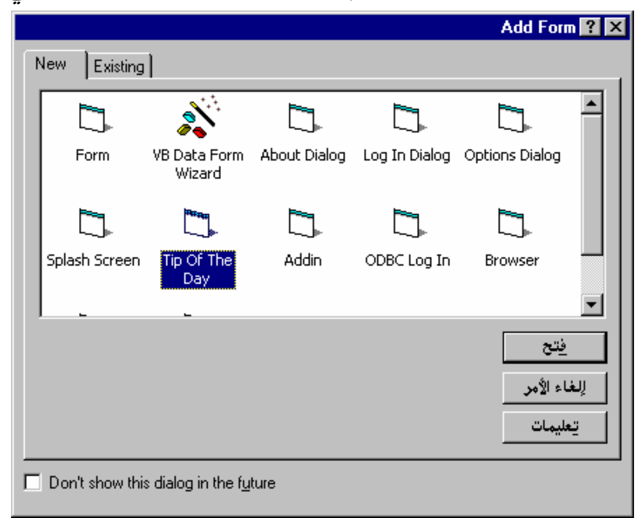
لازالت هناك طريقة أخرى لإنشاء مربعات حوار خاصة، وذلك باستخدام النماذج مسبقة التحضير Prepared Forms، لنشاهد ذلك على أرض الواقع. اتبع الخطوات التالية:

- أنشئ مشروعاً جديداً من النوع التنفيذي القياسي Standard EXE.

لن نطالبك بحفظ هذا المشروع، لأن غرضنا من هذه الخطوات، مجرد توضيح كيفية الوصول إلى النماذج مسبقة التحضير.

- اختر **Add Form** من القائمة **Project**.

*يستجيب فيجول بيسك بإظهار **Add Form** المبين في الشكل التالي:*



الشكل 6-16

الإطار **Add Form**.

يوجد في الشكل 6-16، كما تلاحظ أنواعاً مختلفة من النماذج مسبقة التحضير يمكن الاختيار من بينها.

إذا رجعت في هذا الفصل إلى الوراء قليلاً. ستجد أننا أضفنا النموذج frmGetMonth باختيار البند **Add Form** من القائمة **Project** ثم اخترنا الرمز Form من الإطار Add Form، وبعد ذلك أنهينا تمثيل النموذج بوضع عناصر التحكم في النموذج، وربطها مع النص البرمجي اللازم لإظهار وإخفاء النموذج frmGetMonth.

◻اختر الرمز Tip of the Day من الإطار Add Form، ثم انقر الزر **فتح** في ذلك الإطار.

يستجيب فيجول بيسك بإظهار النموذج Tip of the Day في مرحلة التصميم. (انظر الشكل 6-17):



الشكل 6-17
نموذج تلميح اليوم
في مرحلة التصميم.

ملاحظة

تم تعريب النموذج السابق، حتى أصبح بهذا الشكل، فالنموذج Tip Of The Day الأصلي والذي يأتي مع فيجول بيسك غير معرب أساساً، تستطيع الحصول على النموذج المعرب، من القرص المدمج المرافق لهذا الكتاب في دليل أمثلة الكتاب، الفصل السادس، باسم Tip.frm.

يتضمن فيجول بيسك بعض النماذج مسبقة التحضير التي يسود استخدامها في التطبيقات (مثل النموذج Tip of the Day). تُستأنف عملية التصميم بنفس الطريقة التي اتبعناها عند تصميم النموذج frmGetMonth، والاختلاف الوحيد أن النموذج Tip of the Day. أُعدَّ مسبقاً من أجلك، (وإذا رجعت إلى النص البرمجي له، فستجد جزءاً برمجياً كُتب أيضاً من أجلك).

باختصار لازال عليك الاستمرار في العمل على النموذج (فمثلاً يجب عليك أيضاً بناء النموذج الرئيسي للتطبيق)، ثم كتابة نص البرنامج اللازم لإظهار وإخفاء النموذج أيضاً Tip of the Day تماماً كما فعلنا مع النموذج frmGetMonth .

يعتبر استخدام النماذج مسبقة التحضير، مفيداً أحياناً، فمثلاً تستطيع الاستفادة من النموذج Tip Of the Day الموجود أصلاً عند الحاجة إلى بناء تطبيق ما، (فما الفائدة من

إعادة اختراع العجلات) وطبعاً ستكون أقدر على الاستفادة من النماذج الجاهزة هذه عندما تلم بشكل صحيح بلغة فيجول بيسك، (لأنك ستكون أقدر على فهم النصوص البرمجية المكتوبة لهذه النماذج).

أما في الوقت الحاضر فلن تستخدم النماذج الجاهزة، ولكن بعد انتهاءك من هذا الكتاب والتمرس على بناء تطبيقاتك الخاصة، ارجع إلى الإطار Add Form وحاول بناء بعض التطبيقات التي تستخدم بعض النماذج الجاهزة.

- أنهِ فيجول بيسك، ولا توجد حاجة لحفظ المشروع أو نماذج هذا المشروع.

الخلاصة

شرح هذا الفصل بإسهاب، مربعات الحوار على اختلاف أنواعها. وتعلمت كيفية إظهار رسالة ما للمستخدم، أو سؤال المستخدم عن أمر معين ومعرفة رده ومعالجته. باستخدام العبارة MsgBox أو التابع الوظيفي ()MsgBox.

كما تعلمت كيفية الحصول من المستخدم، على معلومات نصية أو رقمية أو زمنية. باستخدام التابع الوظيفي ()InputBox.

تعلمت أيضاً، كيفية إنشاء مربعات حوار خاصة بك، واستخدامها في برامجك. وعند إنشاء مربعات الحوار الخاصة، فإنه لا يوجد حدود لعدد الأزرار أو عناصر التحكم التي يمكنك وضعها على مربع حوارك الخاص.

غطى هذا الفصل أيضاً مربعات الحوار الشائعة، مثل مربع حوار لون أو مربع حوار فتح أو مربع حوار حفظ، وغيرها من مربعات الحوار الأخرى.

الفصل السابع

عناصر التحكم الرسومية

من بين المظاهر الهامة لاستخدام فيجول بيسك، أنه يسمح لك بسهولة، بإنشاء البرامج التي تحوي رسوماً. سنتعلم في هذا الفصل كيف نكتب برنامجاً يحتوي على عناصر تحكم رسومية.

تعريف وحدة القياس Twip

يمكننا إظهار شتى العناصر الرسومية في فيجول بيسك، مثل الخطوط، والدوائر، والصور النقطية Bitmap، وغير ذلك. ويلزمنا طبعاً تحديد أبعاد هذه العناصر (كطول الخط وقطر الدائرة).

باستطاعة فيجول بيسك استخدام وحدات عديدة، لتحديد مواقع وأبعاد العناصر الرسومية، لكن الوحدة الأكثر شيوعاً، تدعى Twip، وكل بوصة تساوي 1440 Twips.

الألوان Colors

يعتبر اللون من المزايا الهامة لعناصر التحكم الرسومية، تستطيع استخدام التابع الوظيفي RGB() والتابع الوظيفي QBColor() لتحديد لون عنصر التحكم.

تحديد الألوان بواسطة التابع الوظيفي RGB()

يمكّنك التابع الوظيفي RGB() من تحديد الألوان. حيث الأحرف RGB يقصد بها الألوان الرئيسية الثلاثة: الأحمر Red والأخضر Green والأزرق Blue، باعتبار أن كافة الألوان التي يمكن إظهارها على الشاشة، تتولد بالمزج بين هذه الألوان الرئيسية الثلاثة.

للتابع RGB() ثلاثة وسائط: تحدد قيمة الوسيط الأول مقدار اللون الأحمر في اللون النهائي، بينما يمثل الوسيط الثاني مقدار اللون الأخضر في اللون النهائي، وأخيراً، يمثل الوسيط الثالث مقدار اللون الأزرق في اللون النهائي، فمثلاً، نستخدم العبارة التالية لإرجاع اللون الأحمر:

```
BackColor = RGB(255,0,0)
```

القيمة العظمى لكل وسيط في التابع الوظيفي RGB تساوي 255، والقيمة الدنيا للوسيط تساوي الصفر، وهكذا تمثل العبارة RGB (255,0,0) اللون الأحمر، وتمثل العبارة RGB(0,255,0) اللون الأخضر، بينما تمثل العبارة RGB (0,0,255) اللون الأزرق.

استخدم العبارة التالية على سبيل المثال لتبديل الخاصية BackColor لنموذج يدعى frmMyForm إلى اللون الأزرق:

```
frmMyform.BackColor = RGB(0,0,255)
```

لتوليد اللون الأصفر مثلاً، استخدم العبارة RGB(255,255,0)، بينما تولد العبارة RGB(0,0,0) اللون الأسود طبعاً، أما اللون الأبيض فهو فتولده العبارة RGB(255,255,255).

تعتبر الخبرة والممارسة أفضل طريقة للتمكن من استخدام التابع RGB() على الوجه الأمثل.

تحديد الألوان بواسطة التابع الوظيفي QBColor()

يقدم التابع QBcolor() طريقة أخرى سهلة لتحديد اللون، ولهذا الناتج وسيط واحد فقط، يمكن تمثيله بقيمة صحيحة تمتد من الصفر وحتى 15، استخدم العبارة التالية على سبيل المثال لتبديل الخاصية BackCalor لنموذج يدعى frmMyForm إلى اللون الرمادي:

```
frmMyForm.BackColor=QBColor(8)
```

يعطي الجدول 4-3 في الفصل الرابع، قائمة بالألوان الست عشرة المحتملة والقيم المرافقة لهذه الألوان.

يعتبر الناتج الوظيفي QBColor() أسهل استخداماً من سابقه RGB()، إلا أن عدد الألوان التي يقدمها أقل بكثير (16 لوناً فقط).

ملاحظة

قدّم فيجول بيسك التابع QBColor()، بغرض التوافق مع الألوان المستخدمة سابقاً في لغة **Q**uick **B**asic،كما هو واضح من تسمية التابع نفسه.

عنصر تحكم رسم الخط

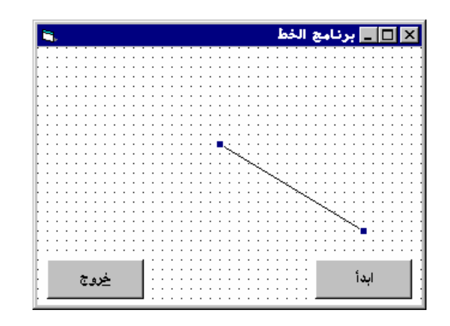
يُستخدم عنصر التحكم Line لرسم الخطوط، يوضح برنامج الخط القادم، كيف يُستخدم هذا العنصر في البرنامج.

التمثيل المرئي لبرنامج الخط

اتبع الخطوات التالية لبناء نموذج برنامج الخط:

- أنشئ الدليل C:\VB5Prg\Ch07، سنستخدم هذا الدليل لحفظ العمل.
- أنشئ مشروعاً جديداً Standard EXE باختيار **New Project** من القائمة **File**.
- احفظ نموذج المشروع بالاسم Line.Frm في الدليل C:\VBb5Prg\Ch07، واحفظ ملف المشروع بالاسم Line.vbp في الدليل C:\VB5Prg\Ch07.
- أنشئ نموذج برنامج الخط، حسب الجدول 7-1.

يبين الشكل 7-1 النموذج المكتمل.



الشكل 7-1
نموذج برنامج الخط.

جدول 7-1. جدول خصائص النموذج frmLine.

الكائن	الخاصية	القيمة
Form	**Name**	**frmLine**
	Caption	برنامج الخط
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdStart**
	Caption	ابدأ
	RightToLeft	True
Line	**Name**	**linLine**
	X1	2760
	X2	3960
	Y1	1800
	Y2	2280

ضع مؤشر الفأرة (بدون نقر) فوق رمز عنصر تحكم الخط، فيظهر مستطيل أصفر يحمل الرسالة Line داخله.

إدخال نص برنامج الخط

سنُدخل الآن نص برنامج الخط:

- تحقق أن قسم التصاريح العامة للنموذج frmLine يحوي العبارة Option Explicit بداخله:

```
'يجب التصريح عن كل المتحولات
```

```
Option Explicit
```

- أدخل النص التالي في الإجراء ()cmdExit_Click:

```
Private Sub cmdExit_Click()
    End
End Sub
```

- أدخل النص التالي في الإجراء ()cmdStart_Click:

```
Private Sub cmdStart_Click()
    'وضع قيم عشوائية لاحداثيات
    'بداية ونهاية الخط
    linLine.X1 = Int(frmLine.Width * Rnd)
    linLine.Y1 = Int(frmLine.Height * Rnd)
    linLine.X2 = Int(frmLine.Width * Rnd)
    linLine.Y2 = Int(frmLine.Height * Rnd)
End Sub
```

- احفظ المشروع، باختيار **Save Project** من القائمة **File**.

تنفيذ برنامج الخط

دعنا نشاهد ما كتبناه قيد التنفيذ:

- نفّذ برنامج الخط.
- انقر زر ابدأ عدة مرات، ولاحظ أن الخط يغيّر طوله وموقعه مع كل نقرة.

بسبب إسناد العنوان &Start إلى الزر ابدأ في الخاصية Caption يمكنك ضغط Alt+S على لوحة المفاتيح بدلاً من النقر بواسطة الفأرة على الزر ابدأ في كل مرة.

كيف يعمل برنامج الخط

يستخدم برنامج الخط الإجراء ()cmdStart_Click لإظهار عنصر التحكم Line في مواقع مختلفة كلما نقر المستخدم على الزر **ابدأ**.

نص الإجراء ()cmdStart_Click

ينفذ هذا الإجراء كلما تم نقر الزر **Start** (أو الضغط على Alt+S):

```
Private Sub cmdStart_Click()
    'وضع قيم عشوائية لاحداثيات
```

```
    'بداية ونهاية الخط
    linLine.X1 = Int(frmLine.Width * Rnd)
    linLine.Y1 = Int(frmLine.Height * Rnd)
    linLine.X2 = Int(frmLine.Width * Rnd)
    linLine.Y2 = Int(frmLine.Height * Rnd)
End Sub
```

يُرجِع التابع الوظيفي Rnd() عدداً عشوائياً بين الصفر والواحد وتُمثِل الخاصية Width عرض النموذج frmLine، وهكذا ينتج عن حاصل جداء عرض النموذج برقم بين الصفر والواحد، ينتج عدد يقع بين الصفر وعرض النموذج:

```
    frmLine.Width*Rnd
```

فإذا أعاد التابع Rnd القيمة صفر، فسيساوي ناتج الجداء الصفر أيضاً، أما إذا أرجع القيمة واحد، فسيساوي الجداء عرض النموذج الكامل، وإذا أعاد التابع Rnd القيمة 0.75 مثلاً، فستساوي النتيجة (3/4) من عرض النموذج.

يحول التابع الوظيفي Int() وسيطه إلى قيمة صحيحة، فمثلاً، تعود العبارة Int(3.5) القيمة 3، وتعود العبارة Int(7.999) القيمة 7. وهكذا فإن ناتج العبارة Int(frmLine.Width * Rnd) يعود بعدد صحيح، يقع بين الصفر وعرض النموذج.

يتم إسناد قيمة صحيحة للخاصية X1 لعنصر التحكم Line، بواسطة العبارة الأولى من الإجراء cmdStart_click(). والقيمة الصحيحة تقع بين الصفر وعرض النموذج:

```
    linLine.X1=Int(frmLine.Width * Rnd)
```

تُمثل الخاصية X1 لعنصر التحكم Line، الإحداثي الأفقي لنقطة بداية الخط، بنظام إحداثيات يعود إلى النموذج (باعتبار أن الخط سيُرسم على نموذج)، يُعرّف نظام الإحداثيات الافتراضي لفيجول بيسك، الإحداثيين X1=0 و Y1=0، باعتبارها الزاوية اليسارية العلوية للنموذج.

أسندت العبارة التالية من الإجراء cmdStart_Click() للخاصية Y1 لعنصر التحكم Line بقيمة صحيحة، والقيمة الصحيحة هذه تقع بين الصفر وارتفاع النموذج:

```
    linLine.Y1=Int(frmLine.Height * Rnd)
```

تمثل الخاصية Y1 لعنصر التحكم Line الإحداثي العمودي لنقطة بداية الخط.

يتم إسناد قيمة صحيحة إلى الخاصية Y2 وX2 من عنصر التحكم، بواسطة العبارتان الأخيرتان في الإجراء cmdStart_Click() وتمثلان إحداثي نقطة نهاية الخط:

```
linLine.X2=Int(frmLine.Width * Rnd)
linLine.X2=Int(frmLine.Height * Rnd)
```

يتحرك الخط عند نقر الزر **ابدأ** إلى مواقع جديدة، اعتماداً على الأرقام العشوائية المتولدة عن توابع Rnd.

أي بكلمة أخرى، يتسبب نقر الزر **ابدأ** بوضع الخط في مواقع عشوائية في نافذة البرنامج.

المزيد من خصائص عنصر تحكم الخط

سنتمرن على مزيد من خصائص العنصر Line أثناء مرحلة التصميم:

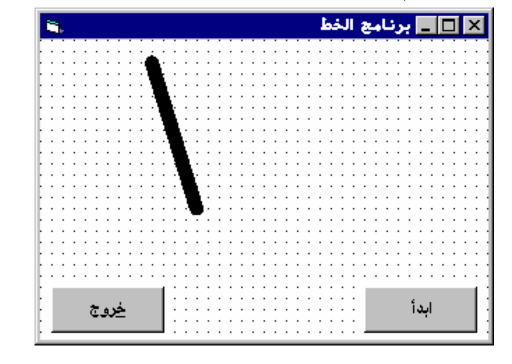
- اختر عنصر التحكم Line أثناء مرحلة التصميم (أي انقل التركيز إليه بنقره بواسطة الفأرة).
- غيّر الخاصية BorderColor لهذا العنصر إلى اللون الأحمر Red.

يستجيب فيجول بيسك بتبديل لون الخط إلى اللون الأحمر.

- غيّر الخاصية BorderWidth لهذا العنصر إلى 10.

يستجيب فيجول بيسك بتبديل عرض الخط إلى 10 Twips.

- احفظ المشروع باختيار **Save Project** من القائمة **File**.
- نفّذ برنامج الخط، وانقر الزر **ابدأ** عدة مرات، يُظهر برنامج الخط الآن، عنصر التحكم Line كخط أحمر بعرض 10 Twips (انظر الشكل 7-2).



الشكل 7-2

عنصر التحكم Line

كخط أحمر بعرض 10 Twips.

- تستطيع أيضاً تبديل لون الخط وعرض الخط أثناء مرحلة التنفيذ.
- أدخل العبارتين التاليتين في بداية الإجراء cmdStart_Click():

```
linLine.BorderColor = RGB(Int(255 * Rnd), _
                          Int(255 * Rnd), _
                          Int(255 * Rnd))
linLine.BorderWidth = Int(100 * Rnd) + 1
```

يتم إسناد قيمة جديدة إلى الخاصية BorderColor بواسطة العبارة الأولى. وهذه القيمة الجديدة هي القيمة المعادة من التابع الوظيفي RGB() حيث كل وسيط من وسائطه الثلاثة، ينتج عن رقم عشوائي بين الصفر و255، وهكذا فالقيمة المعادة من التابع RGB() تمثل لوناً عشوائياً.

يتم إسناد قيمة جديدة إلى الخاصية BorderWidth بواسطة العبارة الثانية. القيمة الجديدة هذه عبارة عن رقم عشوائي يقع بين صفر و100، مع إضافة واحد للنتيجة. أي ((0÷100) +1).

مما يعني أنه يتم إسناد قيمة جديدة بين 1 Twips و 10 Twips إلى خاصية عرض الخط. لاحظ أننا أضفنا واحد إلى العبارة Int(100 * Rnd)، لأن أقل قيمة مقبولة للخاصية BorderWidth لا يجوز أن تساوي الصفر.

- نفّذ برنامج الخط.

*يتبدل لون وعرض الخط مع كل نقرة للزر **ابدأ**. انتبه إلى أنك لن تشاهد الخط أحياناً، عندما يكون اللون العشوائي الناتج، نفس لون أرضية النموذج.*

عنصر تحكم رسم الأشكال

يُستخدم عنصر التحكم هذا، لرسم أشكال عديدة: كالمستطيل والمربع والمستطيل ذو الزوايا المدورة والمربع ذو الزوايا المدورة والدائرة والقطوع الناقصة.

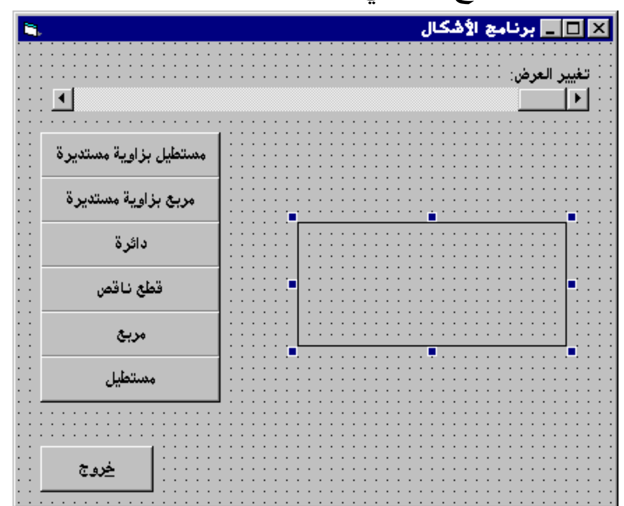
يوضح برنامج الأشكال كيفية إظهار هذه الأشكال على اختلاف أنواعها.

التمثيل المرئي لبرنامج الأشكال

اتبع الخطوات التالية لبناء نموذج برنامج الأشكال:

- أنشئ مشروعاً جديداً من النوع التنفيذي القياسي Standard EXE.
- احفظ نموذج المشروع بالاسم Shape.Frm في ذات الدليل C:\VB5Prg\Ch07، واحفظ ملف المشروع في ذات الدليل بالاسم Shape.vbp.
- أنشئ نموذج برنامج الأشكال تبعاً للجدول 7-2

يظهر النموذج كما في الشكل 7-3.



الشكل 7-3

النموذج frmShape.

الجدول7-2. جدول خصائص برنامج الأشكال.

الكائن	الخاصية	القيمة
Form	**Name**	**frmShape**
	Caption	برنامج الأشكال
	RightToLeft	True
Horizontal Scroll Bar	**Name**	**hsbWidth**
	Min	1
	Max	10

	Value	1
	RightToLeft	True
CommandButton	**Name**	**cmdRndRect**
	Caption	مستطيل بزاوية مستديرة
	RightToLeft	True
CommandButton	**Name**	**cmdRndSqr**
	Caption	مربع بزاوية مستديرة
	RightToLeft	True
CommandButton	**Name**	**cmdCircle**
	Caption	دائرة
	RightToLeft	True
CommandButton	**Name**	**cmdOval**
	Caption	قطع ناقص
	RightToLeft	True
CommandButton	**Name**	**cmdSquare**
	Caption	مربع
	RightToLeft	True
CommandButton	**Name**	**cmdRectangle**

الكائن	**الخاصية**	**القيمة**
	Caption	مستطيل
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
Label	**Name**	**lblInfo**

	Caption	تغيير العرض
	RightToLeft	True
Shape	**Name**	**shpAllShapes**

إدخال نص برنامج الأشكال

سندخل الآن نص برنامج الأشكال:

- تحقق أن قسم التصاريح العامة يحوي العبارة Option Explicit أي:

```
' يجب التصريح عن كل المتغيرات
Option Explicit
```

- أدخل النص التالي داخل الإجراء ()cmdExit_Click:

```
Private Sub cmdExit_Click()
    End
End Sub
```

- أدخل النص التالي داخل الإجراء ()cmdCircle_Click:

```
Private Sub cmdCircle_Click()
    ' تغيير الشكل إلى دائرة
    shpAllShapes.Shape = vbShapeCircle
End Sub
```

- أدخل النص التالي داخل الإجراء ()cmdRectangle_Click:

```
Private Sub cmdRectangle_Click()
    ' تغيير الشكل إلى مستطيل
    shpAllShapes.Shape = vbShapeRectangle
End Sub
```

أدخل النص التالي داخل الإجراء ()cmdOval_Click:

```
Private Sub cmdOval_Click()
    ' تغيير الشكل إلى قطع ناقص
    shpAllShapes.Shape = vbShapeOval
End Sub
```

- أدخل النص التالي داخل الإجراء ()cmdRndRect_Click:

```
Private Sub cmdRndRect_Click()
    ' تغيير الشكل إلى مستطيل بزوايا مستديرة

    shpAllShapes.Shape = vbShapeRoundedRectangle
```

```
End Sub
```

❑أدخل النص التالي داخل الإجراء ()cmdRndSqr_Click:

```
Private Sub cmdRndSqr_Click()
    ' تغيير الشكل إلى مربع بزوايا مستديرة
    shpAllShapes.Shape = vbShapeRoundedSquare
End Sub
```

❑ أدخل النص التالي داخل الإجراء ()cmdSquare_Click:

```
Private Sub cmdSquare_Click()
    ' تغيير الشكل إلى مربع
    shpAllShapes.Shape = vbShapeSquare
End Sub
```

❑أدخل النص التالي داخل الإجراء ()hsbWidth_Scroll:

```
Private Sub hsbWidth_Scroll()
    'تغيير عرض حدود الشكل حسب قيمة شريط الازاحة
    shpAllShapes.BorderWidth = hsbWidth
End Sub
```

❑ أدخل النص التالي داخل الإجراء ()hsbWidth_Scroll:

```
Private Sub hsbWidth_Scroll()
    'تغيير عرض حدود الشكل حسب قيمة شريط الازاحة
    shpAllShapes.BorderWidth = hsbWidth
End Sub
```

❑ احفظ المشروع، باختيار البند **Save Project** من قائمة **File**.

تنفيذ برنامج الأشكال

يظهر برنامج الأشكال كما في الشكل 7–3، عند تنفيذ البرنامج، يتغير الشكل الظاهر تبعاً للزر الذي يتم النقر عليه من قبل المستخدم، فمثلاً يؤدي نقر الزر **دائرة** إلى تحويل الشكل إلى دائرة.

يتغير عرض حدود الشكل، عند تغير موضع شريط التمرير الأفقي، يوضح الشكل 7–4 برنامج الأشكال، بعد نقر الزر **دائرة**، وتغيير موقع مؤشر شريط التمرير.

كيف يعمل برنامج الأشكال

يغيّر نص برنامج الأشكال، الشكل الذي سيظهره عنصر تحكم الأشكال، بالإضافة إلى عرض حدوده، أطلقنا على العنصر التسمية shpAllshapes.

نص الإجراء **cmdRectangle_click()**

يُنفذ الإجراء cmdRectangle_click() عند نقر الزر **مستطيل**. تسند العبارة الوحيدة في هذا الإجراء القيمة صفر للخاصية Shape التابعة لعنصر تحكم الأشكال، (والذي سميناه shpAllShapes) وهي المسؤولة عن تحويل الشكل إلى مستطيل:

```
ShpAllShapes=0
```

وبنفس الطريقة، يُنفذ الإجراء cmdSquare_click() عند نقر الزر **مربع**، ويتم إسناد القيمة واحد إلى الخاصية Shape بواسطة العبارة الوحيدة في هذا الإجراء، وهي المسؤولة عن تحويل الشكل إلى مربع. يُدرج الجدول 7-3 القيم المقبولة للخاصية Shap مع مدلولها.



الشكل 7-4

تبديل الشكل إلى دائرة.

جدول 7-3. القيم الممكنة للخاصية Shape.

القيمة	نوع الشكل
0	مستطيل
1	مربع

2	قطع ناقص
3	دائرة
4	مستطيل بزوايا مستديرة
5	مربع بزوايا مستديرة

نص الإجراء ()hsbWidth_Change

يُنفذ هذا الإجراء عند تبديل موقع مؤشر شريط التمرير، وحسب ما يتبين من جدول خصائص برنامج الأشكال، فإن القيمة الصغرى لهذا الشريط (الخاصية Min) تساوي 1، والخاصية Max تساوي 10، وهي القيمة العظمى، مما يعني أنك تستطيع تغيير الخاصية Value (قيمة مؤشر شريط التمرير) بين الواحد والعشرة.

تُسند هذه القيمة إلى الخاصية BorderWidth للعنصر shpAllShapes كما يلي:

```
ShpAllShapes.BorderWidth= hsbWidth.value
```

وهكذا يتبدل عرض حدود العنصر shpAllShapes تبعاً لموقع مؤشر شريط التمرير.

خصائص أخرى لعنصر التحكم Shape

يقدم برنامج الأشكال خاصيتين فقط من خصائص عنصر التحكم Shape، يمكن التمرن على الخواص الأخرى لهذا العنصر بوضع شكل ما (Shape) في نموذج، ثم تبديل خصائصه.

اتبع الخطوات التالية كإرشادات:

- ضع عنصر تحكم Shape في نموذج.

يستجيب فيجول بيسك بإظهار الشكل كمستطيل افتراضي.

- غير الخاصية Shape إلى دائرة. (القيمة 3).

يستجيب فيجول بيسك بتبديل الشكل من المستطيل إلى دائرة.

- غيّر الخاصية FillColor إلى اللون الأحمر Red.

لا يملأ فيجول بيسك الدائرة بلون أحمر، لأن الخاصية FillStyle للشكل تساوي Transparent (أي شفاف).

- غيّر الخاصية FillStyle إلى Solid (مصمت).

يستجيب فيجول بيسك بملىء الدائرة بلون أحمر.

الصور

يُعتبر عنصرا التحكم Line و Shape، قادرين على رسم الأشكال الهندسية البسيطة كالخطوط والمربعات والدوائر ....الخ. لإظهار الأشكال الأكثر تعقيداً، يلزمك استخدام ملف صورة معين.

يمكن وضع ملفات الصور، على نموذج ما باستخدام عنصر تحكم الرسمة Image، أو عنصر تحكم الصورة Picture.

تُستخدم الخاصية Picture لكلا عنصري التحكم المذكورين لهذا الغرض. فإذا أردت وضع ملف صورة في كائن (Object) سيتوجب عليك تغيير الخاصية Picture لهذا الكائن.

فمثلاً لوضع ملف صورة في عنصر تحكم Image، حدد اسم الملف المقصود في الخاصية Picture لعنصر التحكم Image.

تستطيع تشكيل ملف صورة بواسطة برنامج رسم خاص، وأبسط مثال على برامج الرسم، برنامج الرسام Paint، يمكن استخدام الرسام لتشكيل الصورة، ثم حفظها في ملف بالامتداد BMP، أو يمكن الحصول على صور احترافية جاهزة، يطلق على الصور الجاهزة مصطلح Clip Art.

وضع الصور على نموذج أثناء مرحلة التصميم

تستطيع وضع الصور على نموذج ما، خلال كلا المرحلتين:

- مرحلة التصميم (مرحلة التمثيل المرئي).
- مرحلة التنفيذ (مرحلة كتابة البرنامج).

اتبع الخطوات التالية لتشكيل صورة بواسطة برنامج الرسام:

- شغل برنامج الرسام.

- اختر البند **سمات** (Attributes) من قائمة **صورة** للرسام.

يستجيب برنامج الرسام بإظهار مربع الحوار "سمات".

- اختر واحدة **البوصة**.
- ضع **العرض** على 3.
- ضع **الارتفاع** على 3.
- انقر زر **موافق** لمربع الحوار **سمات**.

يستجيب برنامج الرسام بإظهار صورة خالية بارتفاع ثلاث بوصات وعرض ثلاث بوصات.

- استخدم أدوات الرسام لرسم صورة ما.
- احفظ العمل المنجز في الملف MyPic.bmp في الدليل C:\VB5Prg\Ch07.

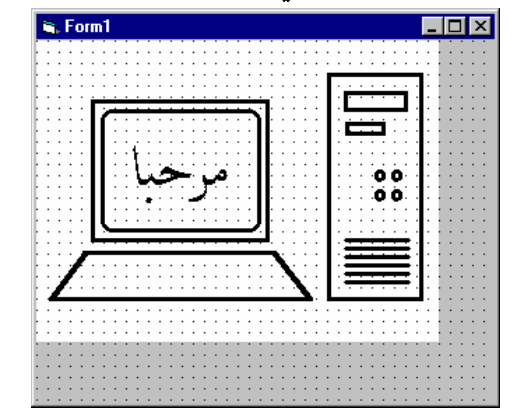
لدينا الآن ملف صورة امتداده bmp ويمكن وضعه على النموذج:

- أنشئ مشروعاً جديداً قياسياً Standard EXE.

يستجيب فيجول بيسك بإظهار نموذج يدعى Form1.Frm.

- حدد الملف MyPic.bmp في الخاصية Picture للنموذج Form1.

تظهر الصورة التي رسمتها للتو، على خلفية النموذج (انظر الشكل 7−5).



الشكل 7−5

وضع صورة تم تشكيلها بواسطة برنامج الرسام على النموذج.

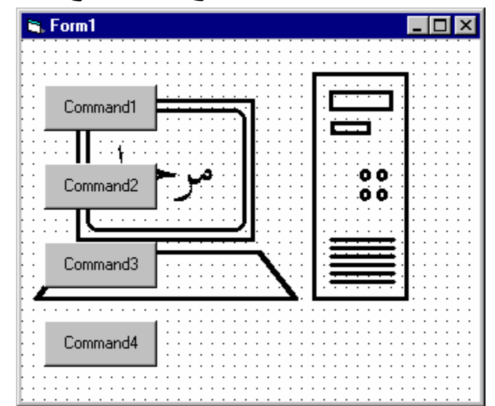
تظهر الصورة على النموذج (كما يظهر من الشكل 7−5)، بدءاً من الإحداثيتين X=0 و Y=0 للنموذج، (أي عند الزاوية اليسرى العليا). ولا تستطيع مد أو تقليص الصورة

التي وضعتها على النموذج، فالصورة تمتلك نفس الأبعاد المحددة في مربع الحوار سمات لبرنامج الرسام.

تخدّم الصورة الموضوعة على النموذج، كخلفية. ولجعل هذه الصورة تلتحم مع الأرضية، قم بما يلي:

- اختر اللون الأبيض للخاصية BackColor للنموذجForm1، على فرض أن خلفية الصورة MyPic.bmp ملونة باللون الأبيض أيضاً، وبهذه الطريقة تلتحم الصورة مع النموذج بشكل تام.
- اسحب حواف النموذج بحيث تظهر كامل الصورة في النموذج Form1، تستطيع وضع أزرار أوامر، أو عناصر تحكم أخرى مباشرةً فوق الصورة.

يبين الشكل 7–6 النموذجForm 1 مع عدة أزرار أوامر موضوعة على النموذج.



الشكل 7–6

وضع أزرار أوامر على الصورة الخلفية للنموذج.

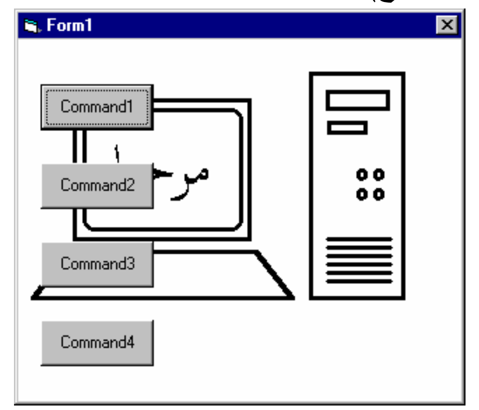
الآن وقد صار للنموذج صورة خلفية، قد ترغب بمنع المستخدم من تكبير أو تصغير النموذج (لأن الصورة لن تكبر أو تصغر مع النموذج).

- ضع الخاصية BorderStyle للنموذج Form1 على 1-FixedSingle.
- نفّذ البرنامج.

حسب ما يبينه الشكل 7–7 يمتلك إطار البرنامج صورة أرضية، (ولا يستطيع المستخدم تكبير أو تصغير الإطار، بسبب إسناد القيمة FixedSingle للخاصية BorderStyle للنموذج Form1.

- اخرج من البرنامج (لا نطالبك بحفظ المشروع فالسبب الوحيد الذي دفعنا لإنشاء

هذا المشروع، تمكينك من مشاهدة عمل الخاصية Picture للنموذج على أرض الواقع).



الشكل 7−7

النموذج مع إسناد القيمة 1-FixedSingle للخاصية BorderStyle.

ملاحظة
يمكن الحصول على صورة سريعة، بحفظ محتويات الشاشة في ملف صورة كالتالي: − في أي وقت تريد حفظ الشاشة الحالية، اضغط المفتاح **Print Screen** على لوحة المفاتيح. (سوف تجد هذا المفتاح أعلى يمين لوحة المفاتيح تقريباً). − شغّل برنامج الرسام من خلال القائمة **ابدأ** − **البرامج** − **البرامج الملحقة** − **الرسام**. − من قائمة برنامج الرسام، اختر البند **تحرير** ثم البند **لصق**. − إذا سألك برنامج الرسام: .. هل تريد تكبير الصورة النقطية؟ جاوب بنعم. كما تلاحظ، لقد ظهرت صورة الشاشة، التي ضغطت المفتاح **Print Screen** عندها. − احفظ هذه الصورة في ملف، من خلال القائمة **ملف**، البند **حفظ**.
وبذلك تكون قد حصلت على ملف صورة الشاشة الحالية.
يمكنك أيضاً، ضغط المفتاح **Alt+Print Screen** لنسخ محتويات النافذة النشطة فقط، وليس كامل مساحة الشاشة.

وضع الصور على النموذج أثناء مرحلة التنفيذ

تستطيع أيضاً وضع الصور على النموذج أثناء تنفيذ البرنامج، والذي نعبر عنه بمرحلة التنفيذ.

استخدم العبارة التالية لتحميل صورة تدعى C:\VB5Prg\OurPic.bmp، ووضعها على نموذج يدعى frmOurForm باستخدام العبارة التالية:

```
frmOurForm.Picture = LoadPicture("C:\VB5Prg\OurPic.bmp")
```

يمكن وضع صورة واحدة فقط على النموذج، في لحظة معينة، ولهذا يستبدل التابع الوظيفي ()LoadPicture، صورة الخلفية الحالية (أي في حال وجود واحدة على النموذج) بالصورة الجديدة.

ملاحظة

استخدم العبارة التالية لمحو (مسح) صورة موجودة مسبقاً على النموذج.

frmOurForm.Pictare= LoadPictare("")

حالما تنفيذ العبارة السابقة يصبح النموذج بدون صورة خلفية.

عنصر التحكم Image

تستطيع أيضاً وضع ملفات الصور BMP في عنصر التحكم Image، الذي يدعم الخاصية Stretch. تمكّنك هذه الخاصية من مد (تكبير) الصورة إلى أي حجم شئت. (لا يدعم النموذج ولا عنصر تحكم Picture هذه الخاصية).

- استخدم العبارة التالية لتحميل الصورة C:\VB5Prg\Ch07\MyPic.bmp ووضعها في عنصر التحكم Image المدعو imgMyImage:

```
imgMyImage.Picture = LoadPicture("C:\VB5Prg\Ch07\MyPic.bmp")
```

يتسبب التابع الوظيفي ()LoadPicture باستبدال الصورة الحالية في حال وجودها حسبما اتفقنا لأن عنصر التحكم Image لا يستوعب أكثر من صورة واحدة في نفس اللحظة.

أما لمحو (مسح) صورة موجودة حالياً في عنصر التحكم Image أثناء مرحلة التنفيذ، فاستخدم العبارة التالية:

```
imgMyImage.Picture = LoadPicture("")
```

استخدم العبارة التالية لإسناد القيمة True إلى الخاصية Stretch للنموذج imgMyImage:

```
imgMyImage.Stretch = True
```

بعد إسناد القيمة True إلى الخاصية Stretch. سوف تكبر الصورة آلياً لتملأ كامل مساحة

عنصر التحكم Imge، أي بكلمة أخرى يكبر فيجول بيسك أو يصغر حجم الصورة آلياً. فمثلاً، إذا كان عنصر التحكم Image بحجم 2 بوصة × 2 بوصة، وكان يحتوي على الصورة MyPic.bmp، فسوف يصغر فيجول بيسك الصورة MyPic.bmp من حجمها الأصلي المساوي إلى 3 بوصة × 3 بوصة، إلى 2 بوصة × 2 بوصة.

تستطيع أيضاً إسناد إحدى القيمتين True أو False، أثناء مرحلة التصميم إلى الخاصية Stretch لعنصر التحكم Image.

عنصر التحكم Picture

يتشابه عنصر التحكم Picture كثيراً مع عنصر التحكم Image، باستثناء أنه يقدّم المزيد من الخصائص، والمزيد من الحوادث والطرق. إلا أنه لا يقدم الخاصية Stretch، (فقط عنصر التحكم Image وحده الذي يدعم الخاصية Stretch).

يقدّم عنصر التحكم Picture الخاصية AutoSize، ويؤدي إسناد القيمة True إلى هذه الخاصية، إلى قيام فيجول بيسك بضبط حجم عنصر التحكم Picture وفقاً لحجم ملف الصورة الذي يحتويها.

فمثلاً، إذا كان ملف الصورة بحجم 3 بوصة × 3 بوصة، فسوف يضبط فيجول بيسك عنصر التحكم Picture بحيث يساوي 3 بوصة × 3 بوصة، لا يقدم النموذج ولا عنصر التحكم Image الخاصية AutoSize.

يستخدم عنصر التحكم Image مصادر أقل Resource، من تلك التي يستخدمها العنصر Picture، ولهذا يتمكن من إعادة رسم الصورة بشكل أسرع.

دمج ملفات الصور في الملفات التنفيذية EXE

اتفقنا إذاً، أن التابع الوظيفي LoadPicture()، يُستخدم لتحميل صورة في عنصر التحكم Image أو في عنصر التحكم Picture أو في النموذج، أثناء مرحلة التنفيذ.

إلا أن استخدام هذا التابع الوظيفي لا يخلو من سلبية، فيجب أن تكون الصورة BMP موجودة في الدليل المحدد ضمن وسيط التابع الوظيفي LoadPicture().

ولهذا يتوجب أن يحتوي القرص الذي توزعه (القرص الذي يحوي برنامجك الكامل) على ملفات الصور التي يستخدمها برنامجك.

من جهة أخرى يصبح ملف الصورة جزءاً من الملف التنفيذي EXE النهائي، إذا تم إسناد ملف الصورة أثناء مرحلة التصميم إلى عنصر التحكم الذي سيحوي الصورة، (النموذج أو عنصر التحكم Image أو العنصر Picture)، فلا تعود هناك حاجة لتوزيع ملف الصورة كملف منفصل.

ملفات الصورة التي يدعمها فيجول بيسك هي: ملفات الصور النقطية BMP، وملفات الأيقونات ICO وملفات الرسوم المتجهية WMF وملفات مؤشرات الفأرة CUR.

ملفات الرسوم النقطية Bitmap Files

هي عبارة عن ملفات تحمل أحد الامتدادين BMP أو DIB. يحتوي ملف الرسوم النقطية على بايتات تصف مواقع وألوان النقاط الضوئية (Pixels) للصورة.

ملفات الأيقونات Icon Files

عبارة عن ملفات تحمل الامتداد ICO. تتشابه هذه الملفات مع الملفات من النوع BMP و DIB، إلا أنها تمثل صوراً يبلغ الحجم الأعظمي لها 32×32 نقطة ضوئية، (النقطة الضوئية Pixel: هي أصغر عنصر نقطي يمكن للشاشة إظهاره، وتتحدد دقة الشاشة به).

ملفات الرسوم المتجهية Meta Files

عبارة عن ملفات تحمل الامتداد WMF، وتحوي لائحة من التعليمات الرسومية التي تصف كيفية توليد الصورة.

ملفات مؤشرات الفأرة Cursor Files

ملفات تحمل الامتداد CUR. تتشابه مع ملفات الأيقونات ICO. وهي ملفات صغيرة تُستخدم عادة لتمثيل مؤشر الفأرة، كالساعة الرملية ورأس السهم.

برنامج العين المتحركة

يمكننا نقل وتحريك عنصر تحكم ما إما بتغيير الخاصيتين Left و Top، أو باستخدام الطريقة Move.

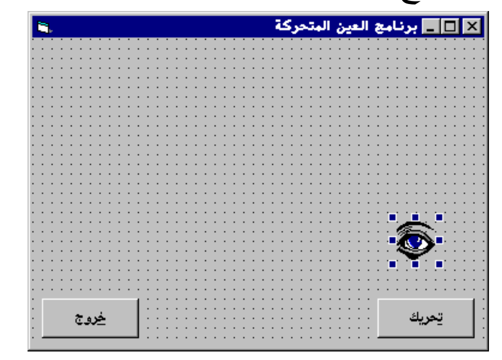
سنكتب برنامجاً يدعى العين المتحركة، يوضح كيف تستطيع تحريك كائن ما، بتغيير الخاصيتين Left و Top لهذا الكائن.

التمثيل المرئي لبرنامج العين المتحركة

سنشرع كعادتنا بالتمثل المرئي للبرنامج:

- أنشئ مشروعاً تنفيذياً قياسياً Standard EXE جديداً.
- احفظ نموذج المشروع بالاسم MoveEye.frm في الدليل C:\VB5prg\Ch07، واحفظ ملف المشروع بالاسم MoveEye.vbp في نفس الدليل.
- أنشئ نموذج برنامج العين المتحركة طبقاً للجدول 7-4.

النموذج المكتمل سيبدو مشابهاً لذاك المبين في الشكل 7-8.



الشكل 7-8

النموذج frmMoveEye

بعد انتهائه في مرحلة التصميم.

جدول 7-4. جدول خصائص النموذج frmMoveEye.

الكائن	الخاصية	القيمة
Form	Name	frmMoveEye

	Caption	برنامج العين المتحركة
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdMove**
	Caption	&تحريك
	RightToLeft	True
Image	**Name**	**imgEye**
	Picture	Eye.Ico
	Stretch	True

يطالبك الجدول 7-4 بإسناد اسم الملف Eye.Ico للخاصية Picture لعنصر التحكم imgEye.

يمكنك العثور على هذا الملف في الدليل الفرعي \Icons\Misc الذي أنشئ عند تنصيب فيجول بيسك. إذا لم يحتو الدليل على هذا الملف، تستطيع استخدام أي ملف بديل.

إدخال نص برنامج العين المتحركة

سنكتب الآن نص برنامج العين المتحركة:

- تحقق أن قسم التصاريح العامة للنموذج frmMoveEye يحتوي عبارة Option Explicit:

```
'يجب الاعلان عن كل المتغيرات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء ()cmdExit_Click:

```
Private Sub cmdExit_Click()
    End
End Sub
```

- أدخل النص التالي ضمن الإجراء ()cmdMove_Click:

```
Private Sub cmdMove_Click()
    Dim Counter As Integer
    'تنفيذ الحلقة 100 مرة
```

```
    For Counter = 1 To 100
      'ازاحة الصورة 20 وحدة إلى اليسار
      imgEye.Left = imgEye.Left - 20
      'انقاض ارتفاع الصورة 20 وحدة منطقية
      imgEye.Top = imgEye.Top - 20
    Next
End Sub
```

تنفيذ برنامج العين المتحركة

لنشاهد ما كتبناه على أرض الواقع:

- نفذ برنامج العين المتحركة.

تتحرك صورة العين عند نقر الزر **تحريك**، وتستطيع تكرار النقر على هذا الزر إلى أن تختفي العين عن الأنظار.

كيف يعمل برنامج العين المتحركة

يحرّك برنامج العين المتحركة، عنصر التحكم Image (صورة العين)، بتبديل الخاصيتين Left و Top.

نص الإجراء cmdMove_Click()

يُنفذ الإجراء cmdMove_Click() آلياً عند نقر الزر **تحريك**:

```
Private Sub cmdMove_Click()
    Dim Counter As Integer
    'تنفيذ الحلقة 100 مرة
    For Counter = 1 To 100
      'ازاحة الصورة 20 وحدة إلى اليسار

      imgEye.Left = imgEye.Left - 20
      'انقاض ارتفاع الصورة 20 وحدة منطقية
      imgEye.Top = imgEye.Top - 20
    Next
End Sub
```

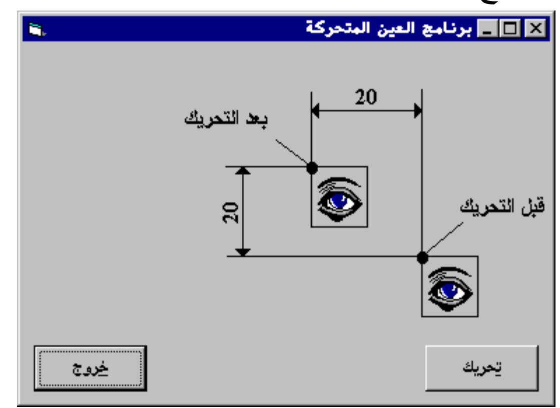
تُنقص العبارة الأولى في الحلقة For الإحداثي العمودي للزاوية اليسرى العليا للصورة بمقدار 20 Twips:

```
imgEye.Top = imgEye.Top - 20
```

كما تنقص العبارة الثانية في الحلقة For الإحداثي الأفقي للزاوية اليسرى العليا للصورة بمقدار 20 Twips أيضاً:

```
imgEye.Left = imgEye.Left - 20
```

يوضح الشكل 7-9 تأثير هاتين العبارتين:



الشكل 7-9

تحريك الصورة بمقدار 20 Twips إلى الأعلى و 20 Twips إلى اليسار.

تُنفذ الحلقة For.Next مائة مرة ولهذا تُحرك الصورة 100 مرة، مما يعطي الناظر وهم الحركة المستمرة.

تحريك أو نقل عنصر تحكم باستخدام الطريقة Move

ينقل برنامج العين المتحركة الذي تناولناه سابقاً الصورة بتبديل الخصائص Top و Left. تعتبر الطريقة Move طريقة أخرى لتحريك عنصر التحكم.

- استبدل نص الإجراء cmdMove_Click() السابق بالنص التالي:

```
Private Sub cmdMove_Click()

    Dim Counter As Integer
    Dim LeftEdge As Single
    Dim TopEdge As Single

    ' وضع احداثيات الصورة وأبعادها في المتحولات
    LeftEdge = imgEye.Left
```

```
    TopEdge = imgEye.Top

    'تنفيذ الحلقة 100 مرة
    For Counter = 1 To 100
      'ازاحة الصورة 20 وحدة إلى اليسار
      LeftEdge = LeftEdge - 20
      'انقاض ارتفاع الصورة 20 وحدة منطقية
      TopEdge = TopEdge - 20
      ' زيادة العرض 10 وحدات منطقية
      imgEye.Move LeftEdge, TopEdge
    Next
End Sub
```

يُجهز هذا الإجراء متحولين هما: LeftEdge و TopEdge، بإسناد الموقع الحالي للزاوية اليسرى العليا للصورة لهما، ثم ينفذ الحلقة For.Next مائة مرة.

تنقص قيمة المتحولين LeftEdge و TopEdge بمقدار 20 Twips مع كل تكرار للحلقة.

تستخدم الطريقة Move مع المتحول LeftEdge باعتباره الإحداثي الأفقي الجديد، والمتحول TopEdge باعتباره الإحداثي العمودي الجديد.

المزيد عن طريقة Move

الصيغة الكاملة لعبارة Move تعطى بالشكل التالي:

```
[Object name].Move newLeft, newTop, newWidth, newHight
```

نستنتج من العبارة السابقة، أن بإمكاننا تعيين عرض وارتفاع جديدين للكائن بعد تحريكه. وإليك المثال التالي:

- استبدل نص الإجراء cmdMove_Click() السابق بالنص التالي:

```
Private Sub cmdMove_Click()

    Dim Counter As Integer
    Dim LeftEdge As Single
    Dim TopEdge As Single
    Dim NewWidthOfImage As Single
    Dim NewHeightOfImage As Single
    imgEye.Stretch = True
    ' وضع احداثيات الصورة وأبعادها في المتحولات
```

```
LeftEdge = imgEye.Left
TopEdge = imgEye.Top
NewWidthOfImage = imgEye.Width
NewHeightOfImage = imgEye.Height
'تنفيذ الحلقة 100 مرة
For Counter = 1 To 100
  'ازاحة الصورة 20 وحدة إلى اليسار
  LeftEdge = LeftEdge - 20
  'انقاض ارتفاع الصورة 20 وحدة منطقية
  TopEdge = TopEdge - 20
  ' زيادة العرض 10 وحدات منطقية
  NewWidthOfImage = NewWidthOfImage + 10
  ' زيادة الارتفاع 10 وحدات منطقية
  NewHeightOfImage = NewHeightOfImage + 10
  imgEye.Move LeftEdge, TopEdge, _
              NewWidthOfImage, NewHeightOfImage
Next
End Sub
```

□نفّذ البرنامج، ولاحظ أن العين تتحرك وتكبر في نفس الوقت، وبعد مائة مرة، يبدو النموذج كما في الشكل 7-10.



الشكل 7-10

استخدام طريقة Move لتحريك وتكبير العين في نفس الوقت.

أول عبارة بعد عبارات التصاريح هي العبارة التالية:

```
imgEye.Stretch = True
```

يتم إسناد القيمة True للخاصية Stretch للعنصر imgEye، بواسطة العبارة السابقة، مما يعني أن البرنامج يستطيع تكبير الصورة (أنجزنا هذه الخطوة مسبقاً أثناء مرحلة

التصميم، إلا أن تكرارها هنا ما هو إلا بهدف توضيح أن الخاصية Stretch يمكن تحديدها أثناء مرحلة التنفيذ أيضاً).

يتم إسناد قيمتين ابتدائيتين تمثلان عرض وارتفاع الصورة، إلى المتحولين NewWidthOfImage و NewHeightOfImage:

```
NewWidthOfImage = imgEye.Width
NewHeightOfImage = imgEye.Height
```

تنفذ الطريقة Move بعد ذلك 100 مرة، ومع كل تكرار للحلقة تزداد قيمة المتحولين NewWidthOfImage و NewHeightOfImage بمقدار 10 Twips.

تحدد الطريقة Move الإحداثيات الجديدة للزاوية اليسرى العليا للصورة، وكذلك العرض والارتفاع الجديدين بعد إنجاز عملية التحريك:

```
imgEye.Move LeftEdge, TopEdge, _
            NewWidthOfImage, NewHeightOfImage
```

ملاحظة

تُستخدم طريقة Move لتحريك أي كائن Object باستثناء القوائم، وقد تكون بعض عناصر التحكم غير مرئية (مثال، عنصر التحكم Timer)، ولهذا لا يحمل تحريكها أي معنى.

مقارنة تقنيات التحريك

ستجد عند مقارنة أوامر برنامج العين المتحركة، باستخدام كلتا تقنيتي التحريك، أن الطريقة Move أفضل طرق التحريك. فاستخدام الخاصتين Top و Left في التحريك يعطي انتقالاً غير متوازن. لن تشعر بهذا الأمر طبعاً عند استخدام حاسب ذي معالج سريع (بنتيوم بسرعة 233 ميغا هرتز) لأن الصورة التي تحركها صغيرة جداً.

ولكن ستلاحظ عند استخدام طريقة Move بعض الوميض على الشاشة. ويزداد هذا الوميض مع تزايد حجم الصورة. هذا بغض النظر عن سرعة الحاسب PC المستخدم هنا. يمكن إزالة هذا الوميض باستخدام تقنيات DirectX.

نقل عنصر التحكم Picture

□أوضح البرنامج السابق أن استخدام طريقة Move لتحريك عنصر تحكم، يعطي حركة أخف من تبديل الخاصيتين Top و Left. يمكنك الحصول على نتائج أفضل باستخدام عنصر التحكم Picture بدل عنصر التحكم Image. ولرؤية كيف تستخدم عنصر التحكم Picture، بدل النموذج frmMoveEye:

□احذف عنصر التحكم Image المسمى imgEye من النموذج باختياره، ثم ضغط المفتاح Delete على لوحة المفاتيح.

□انقر نقراً مزدوجاً على عنصر التحكم Picture فوق مربع الأدوات ليتوضع على النموذج.

□ أسند الاسم picEye إلى الخاصية Name لعنصر التحكم Picture.

□ أسند اسم الملف Eye.Ico إلى الخاصية Picture للعنصر picEye.

□أسند القيمة 0-None للخاصية BorderStyle للعنصر picEye، مما يعني عدم إحاطة الصورة بحدود.

□أسند القيمة 0-Flat للخاصية Appearance للعنصر picEye، مما يعني أن عنصر التحكم هذا سيظهر مسطحاً، وليس ثلاثي الأبعاد.

□أسند القيمة True للخاصية AutoSize للعنصر picEye، لتمكين عنصر التحكم من تغيير حجمه ليتسع صورة العين آلياً.

□أسند اللون الرمادي Gray للخاصية BackColor للعنصر picEye، (حتى تكون خلفية صورة العين، نفس لون خلفية النموذج).

□ استبدل الإجراء cmdMove_Click() السابق بالإجراء التالي:

```
Private Sub cmdMove_Click()
    Dim Counter As Integer
    Dim LeftEdge As Single
    Dim TopEdge As Single
    ' وضع احداثيات الصورة وأبعادها في المتحولات
    LeftEdge = imgEye.Left
    TopEdge = imgEye.Top
```

```
    'تنفيذ الحلقة 100 مرة
    For Counter = 1 To 100
      'ازاحة الصورة 20 وحدة إلى اليسار
      LeftEdge = LeftEdge - 20
      'انقاض ارتفاع الصورة 20 وحدة منطقية
      TopEdge = TopEdge - 20
      imgEye.Move LeftEdge, TopEdge
    Next
End Sub
```

- احفظ المشروع باختيار **Save Project** من القائمة **File** لفيجول بيسك.
- نفّذ البرنامج وانقر الزر **Move**، وراقب كيف تتحرك العين.

مصفوفة عناصر التحكم

يعرّف عنصر التحكم على أنه مصفوفة، وبنود هذه المصفوفة هي عناصر تحكم. فنستطيع مثلاً إنشاء مصفوفة من عناصر التحكم Image.

برنامج القمر

سنكتب الآن برنامجاً يدعى برنامج القمر، وهذا البرنامج سيستخدم مصفوفة من عناصر التحكم Image.

التمثيل المرئي لبرنامج القمر

سنشرع الآن بطور التمثيل المرئي لنموذج برنامج القمر:

- أنشئ مشروعاً تنفيذياً قياسياً Standard EXE جديداً.
- احفظ نموذج المشروع بالاسم Moon.Frm في الدليل C:\VB5Prg\Ch07، واحفظ المشروع بالاسم Moon.vbp في ذ ات الدليل.
- أسند الخصائص التالية للنموذج Moon.Frm:

```
Name:          frmMoon
Caption:       برنامج القمر
RightToLeft:   True
```

- ضع عنصر تحكم Image في النموذج frmMoon، بالنقر المزدوج على رمز

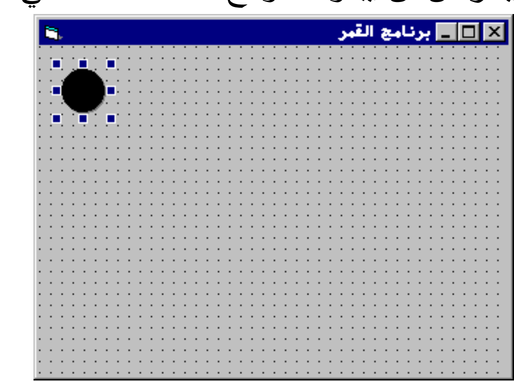
عنصر التحكم Image في إطار مربع الأدوات.

يستجيب فيجول بيسك بوضع عنصر التحكم Image في النموذج frmMoon.

- اسحب عنصر التحكم Image إلى القسم العلوي اليساري من النموذج.
- أسند الاسم imgMoon للخاصية Name لعنصر التحكم Image الذي وضعته في النموذج، (أطلقنا على عنصر التحكم هذا الاسم imgMoon).
- أسند القيمة False للخاصية Visible لعنصر التحكم imgMoon.
- أسند القيمة False للخاصية Stretch لعنصر التحكم imgMoon.
- أسند الملف Moon01.Ico إلى الخاصية Picture لعنصر التحكم imgMoon (يقع هذا الملف ضمن الدليل الفرعي Icons\Elements للدليل الذي ثبّت فيه فيجول بيسك).

طبعاً تستطيع استخدام أي ملف آخر مكان هذا الملف في حال عدم توافره.

يفترض أن يبدو النموذج frmMoon كما في الشكل 7-11.



الشكل 7-11

العنصر الأول في مصفوفة عنصر التحكم Image.

انتهينا الآن من وضع عنصر التحكم Image في النموذج. يعتبر هذا العنصر أول بند (البند ذي الرقم صفر) في مصفوفة العناصر. إلا أن العنصر الذي وضعناه هو حتى هذه اللحظة عنصر تحكم نظامي (أي ليس عنصراً في مصفوفة). وتستطيع التحقق من ذلك بالرجوع إلى الخاصية Index للعنصر imgMoon، والتي ستكون فارغة.

اتبع الخطوات التالية لوضع عنصر تحكم ثاني في مصفوفة عناصر التحكم:

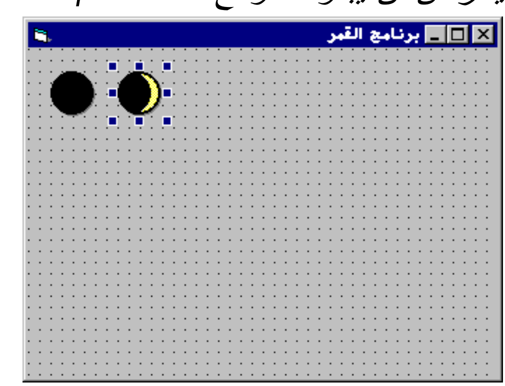
- انقر نقراً مزدوجاً على رمز عنصر التحكم Image في إطار مربع الأدوات.

يستجيب فيجول بيسك بوضع عنصر تحكم Image ثان في النموذج frmMoon،

يحتوي النموذج الآن على عنصري تحكم Image.

- اسحب عنصر التحكم Image الثاني إلى يمين العنصر الأول.
- أسند القيمة False إلى الخاصية Visible للعنصر الثاني.
- أسند القيمة False إلى الخاصية Stretch للعنصر الثاني.
- أسند القيمة Moon 02.Ico إلى الخاصية Picture للعنصر الثاني.

يفترض أن يبدو النموذج frmMoon كما هو مبين في الشكل 7-12.



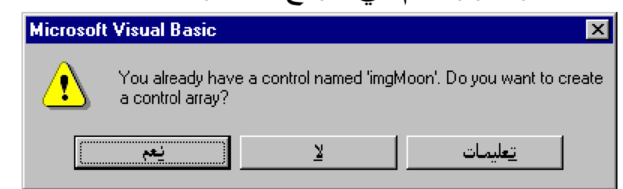
الشكل 7-12

العنصرين الأول والثاني في مصفوفة عناصر التحكم Image.

- أسند الاسم imgMoon إلى الخاصية Name لعنصر التحكم Image الثاني، و انقر الآن في أي مكان من النموذج.

يستجيب فيجول بيسك بإظهار مربع الحوار المبين في الشكل 7-13، حيث يتحقق فيجول بيسك بواسطته، من رغبتك بتسميته العنصر الثاني بنفس اسم عنصر التحكم الأول.

- انقر الزر **نعم** في مربع الحوار المبين بالشكل 7-13.



الشكل 7-13

إنشاء مصفوفة عناصر تحكم Control Array.

لقد أعلمت فيجول بيسك بأنك ترغب بإنشاء مصفوفة عناصر تحكم Control Array.

- تَفَحَّص الخاصية Index لأول عنصري تحكم.

ستجد أن الخاصية Index لعنصر التحكم Image الأول تساوي الصفر، وتساوي 1 لعنصر التحكم Image الثاني.

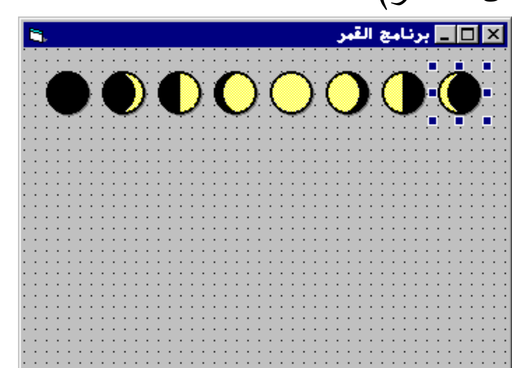
مما يعني أنك تملك الآن مصفوفة عناصر تحكم يدعى ()imgMoon، العنصر الأول في المصفوفة هو (0)imgMoon (وهو الملف Moon01.Ico)، والذي يمثل أول عنصر وضعته في النموذج.

والعنصر الثاني في المصفوفة هو العنصر (1)imgMoon ويمثل الملف Moon02.Ico.

□كرر الآن المعالجة السابقة وأضف ستة عناصر أخرى إلى مصفوفة عناصر التحكم Control Array. سيكون لديك عند الانتهاء، ثمانية عناصر في المصفوفة، حسب ما يبينه الشكل 7-14 والجدول 7-5.

لا تنس إسناد نفس القيم لخصائص العناصر الست المتبقية،(Name و Stretch و Visible).

لاحظ أن فيجول بيسك لا يعيد تنبيهك بإظهار مربع الحوار المبين بالشكل 7-13 عند إضافة العناصر الإضافية إلى المصفوفة ()imgMoon لأن فيجول بيسك علم أن imgMoon عبارة عن مصفوفة عناصر تحكم Control Array (كبر النموذج إذا دعت الحاجة، ليتسع كل الأقمار).



الشكل 7-14

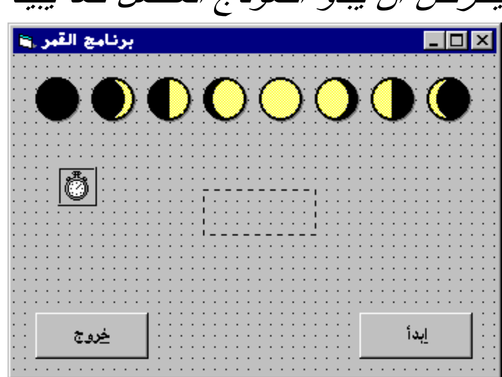
الأقمار الثمانية لمصفوفة عناصر التحكم.

الجدول 7-5. مصفوفة عناصر التحكم المدعو imgMoon().

عنصر المصفوفة	محتويات العنصر
imgMoon(0)	Moon01.Ico
imgMoon(1)	Moon02.Ico
imgMoon(2)	Moon03.Ico
imgMoon(3)	Moon04.Ico
imgMoon(4)	Moon05.Ico
imgMoon(5)	Moon06.Ico
imgMoon(6)	Moon07.Ico
imgMoon(7)	Moon08.Ico

❑ استأنف بناء النموذج frmMoon تبعاً للجدول 7-6.

يفترض أن يبدو النموذج المكتمل كما يبينه الشكل 7-15.



الشكل 7-15

النموذج frmMoon.

الجدول 7–6. جدول خصائص برنامج القمر.

الكائن	الخاصية	القيمة
Form	**Name**	**frmMoon**
	Caption	برنامج القمر
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdStart**
	Caption	ا&بدأ
	RightToLeft	True
Timer	**Name**	**tmrTimer**
	Interval	250
	Enabled	True
Image	**Name**	**imgCurrentMoon**
	Visible	True
	Picture	(None)
	Stretch	False
Image Control	(انظر الجدول 7–5)	(انظر الجدول 7–5)

ملاحظة

اترك الخاصية Picture للعنصر imgCurrentMoon على حالها دون تغيير القيمة الافتراضية لها. لأننا سنسند لها قيمة من ضمن نص البرنامج. لا يعتبر عنصر التحكم imgCurrentMoon جزءاً من مصفوفة عناصر التحكم ()imgMoon.

لا يظهر عنصر التحكم Timer خلال مرحلة التنفيذ، لذلك ضعه على النموذج في أي موقع.

إدخال نص برنامج القمر

سندخل الآن نص برنامج القمر:

- أدخل النص التالي ضمن قسم التصاريح العامة للنموذج frmMoon:

```
'يجب التصريح عن كل المتحولات
Option Explicit
'التصريح هنا عن أي متحول يجعله متاحا في
' frmMoon جميع أجزاء النموذج
Dim gRotateFlag  As Integer
Dim gCurrentMoon As Integer
```

- أدخل النص التالي ضمن الإجراء ()cmdExit_click:

```
Private Sub cmdExit_Click()
    End
End Sub
```

- أدخل النص التالي ضمن الإجراء ()cmdSart_click:

```
Private Sub cmdStart_Click()
    ' التبديل بين حالتي البدء والتوقف
    If gRotateFlag = 0 Then
      gRotateFlag = 1
      cmdStart.Caption = "&توقف"
    Else
      gRotateFlag = 0
      cmdStart.Caption = "&ابدأ"
    End If
End Sub
```

- أدخل النص التالي ضمن الإجراء ()Form_Load:

```
Private Sub Form_Load()
    'إعطاء قيمة أولية للمتحولات
    gRotateFlag = 0
    gCurrentMoon = 0
End Sub
```

- أدخل النص التالي ضمن الإجراء ()tmrTimer_Timer:

```
Private Sub tmrTimer_Timer()
    'سوف ينفذ هذا الإجراء كل ربع ثانية
    If gRotateFlag = 1 Then
```

```
  imgCurrentMoon.Picture = _
          imgMoon(gCurrentMoon).Picture
  gCurrentMoon = gCurrentMoon + 1
  If gCurrentMoon = 8 Then
    gCurrentMoon = 0
  End If
End If
```

```
End Sub
```

- احفظ المشروع باختيار **Save Project** من القائمة **File** لفيجول بيسك.

تنفيذ برنامج القمر

لنشاهد برنامج القمر الآن، وهو يعمل:

- نفّذ برنامج القمر.
- انقر الزر **ابدأ**.
- كما تلاحظ، يظهر القمر وكأنه يدور حول نفسه (محوره).
- انقر الزر **خروج** لإنهاء برنامج القمر.

نص برنامج القمر

يستخدم نص برنامج القمر، مصفوفة من عناصر التحكم Images لإظهار عناصر المصفوفة الواحد تلو الآخر، مما يعطي الإحساس بأن القمر يدور حول نفسه.

نص قسم التصاريح العامة

يصرح قسم التصاريح العامة عن متحولين صحيحين: gRotateFlag و gCurrentMoon. وبذلك يصبح هذان المتحولان مرئيين من قبل كافة إجراءات النموذج frmMoon.

نص الإجراء Form_Load()

ينفذ الإجراء Form_Load() آلياً، عند تشغيل البرنامج،ويعتبر مكاناً مناسباً لإنجاز مختلف مهام الابتداء. فمثلاً يتم إسناد القيمة الابتدائية صفر إلى المتحولين اللذين تم التصريح عنهما في قسم التصاريح العامة:

```
gRotateFlag = 0
```

```
    gCurrentMoon = 0
```

ملاحظة

يُسند فيجول بيسك آلياً قيمة ابتدائية تساوي الصفر للمتحولات عند التصريح عنها، إلا أن إسناد قيم ابتدائية إلى المتحولات في الإجراء Form_Load() يسهّل قراءة نص البرنامج.

نص الإجراء cmdSart_click()

يُنفذ الإجراء cmdSart_click() عند النقر على **ابدأ**. وهو ينفذ بدوره العبارة If.Else. التي تتحقق من القيمة الراهنة للمتحول gRotateFlag:

```
Private Sub cmdStart_Click()
    ' التبديل بين حالتي البدء والتوقف
    If gRotateFlag = 0 Then
      gRotateFlag = 1
      cmdStart.Caption = "&توقف"
    Else
      gRotateFlag = 0
      cmdStart.Caption = "&ابدأ"
    End If
End Sub
```

فإذا كانت القيمة الراهنة للمتحول gRotateFlag تساوي الصفر، تُبدّل هذه القيمة إلى الواحد، ويحوّل عنوان الزر cmdStart إلى **&توقف**. أما إذا كانت القيمة الراهنة لذلك المتحول (أي gRotateFlag) تساوي الواحد، فإن الإجراء يبدلها إلى صفر، ويرجع عنوان الزر cmdStart إلى **ا&بدأ** مرة أخرى.

يُستخدم المتحول gRotateFlag في إجراء الميقاتية الذي سنناقشه للتو.

نص الإجراء tmrTimer_Timer()

يُنفَّذ الإجراء tmrTimer_Timer() كل 250 ميلي ثانية، بسبب إسناد القيمة 250 إلى الخاصية Interval للميقاتية trmTimer.

```
Private Sub tmrTimer_Timer()
    'سوف ينفذ هذا الإجراء كل ربع ثانية
    If gRotateFlag = 1 Then
      imgCurrentMoon.Picture = _
              imgMoon(gCurrentMoon).Picture
      gCurrentMoon = gCurrentMoon + 1

      If gCurrentMoon = 8 Then
        gCurrentMoon = 0
      End If
    End If

End Sub
```

تبقى قيمة المتحول gRotateFlag مساوية إلى الصفر، ما لم تنقر الزر **ابدأ**، ولا تنفذ العبارات الواقعة ضمن كتلة الشرط:

```
If gRotateFlag = 1 Then
  imgCurrentMoon.Picture = _
          imgMoon(gCurrentMoon).Picture
  gCurrentMoon = gCurrentMoon + 1
  If gCurrentMoon = 8 Then
    gCurrentMoon = 0
  End If
End If
```

أما عند نقر الزر **ابدأ**، فتصبح قيمة gRotateFlag مساوية الواحد، عندها تنفذ العبارات الواقعة ضمن كتلة الشرط:

```
If gRotateFlag = 1 Then
  imgCurrentMoon.Picture = _
          imgMoon(gCurrentMoon).Picture
  gCurrentMoon = gCurrentMoon + 1
  If gCurrentMoon = 8 Then
    gCurrentMoon = 0
  End If
End If
```

يعمل جزء البرنامج الموجود ضمن عبارة الشرط على إسناد الخاصية Picture لمصفوفة عناصر التحكم إلى الخاصية Picture للعنصر imgCurrentMoon:

```
imgCurrentMoon.Picture = imgMoon(gCurrentMoon).Picture
```

على سبيل المثال، إذا كانت قيمة المتحول gCurrentMoon مساوية إلى الصفر، فسوف يتم إسناد الخاصية Picture للعنصر الأول (ذي الترتيب صفر) من عناصر مصفوفة التحكم إلى الخاصية Picture للعنصر imgCurrentMoon. مما يعني أن صورة عنصر التحكم الأول Moon01.Ico سوف تظهر (لأن imgMoon(0).Picture يحوي الصورة Moon01.Ico).

تزيد العبارة التالية في كتلة الشرط if قيمة المتحول gCurrentMoon بمقدار واحد:

```
gCurrentMoon = gCurrentMoon + 1
```

وعند التنفيذ التالي للإجراء tmrTimer_Timer() (أي بعد مرور 250 ميلي ثانية)، يشير المتحول gCurrentMoon إلى العنصر التالي في مصفوفة عناصر التحكم.

تختبر العبارات التالية في الإجراء tmrTimer_Timer() قيمة gCurrentMoon، فإذا كانت مساوية إلى 8، فهذا يعني أنه تم للتو إظهار العنصر الثامن من عناصر مصفوفة عناصر التحكم، ولهذا يجب إعادة تصفير المتحول gCurrentMoon، أي:

```
If gCurrentMoon = 8 Then
   gCurrentMoon = 0
End If
```

التحريك Animation

يوضح برنامج القمر مقدار السهولة التي يمكن بها كتابة برنامج تحريك Animation في فيجول بيسك. نستطيع تحسين برنامج القمر بحيث تغدو أوامر التحريك فيه أعمق أثراً.

تحسين برنامج القمر

طوِّر برنامج القمر وفق ما يلي:

- أسند القيمة 55 إلى الخاصية Interval للميقاتية tmrTimer.
- أضف النص البرمجي التالي إلى قسم التصاريح العامة للنموذج frmMoon:

```
Option Explicit
Dim gRotateFlag  As Integer
```

```
    Dim gCurrentMoon As Integer
    Dim gDirection As Integer
    Dim gLeftCorner As Single
    Dim gTopCorner As Single
    Dim gWidthOfMoon As Single
    Dim gHeightOfMoon As Single
    Dim gEnlargeShrink As Integer
```

◻عدّل الإجراء ()Form_Load حيث يغدو بالشكل التالي:

```
Private Sub Form_Load()
    gRotateFlag = 0
    gCurrentMoon = 0
    gDirection = 1
    gLeftCorner = imgCurrentMoon.Left
    gTopCorner = imgCurrentMoon.Top
    gWidthOfMoon = 1
    gHeightOfMoon = 1
    gEnlargeShrink = 1

    imgCurrentMoon.Stretch = True
End Sub
```

◻عدل الإجراء ()tmrTimer_Timer بحيث يغدو كما يلي:

```
Private Sub tmrTimer_Timer()
    'سوف ينفذ هذا الإجراء كل ربع ثانية
    If gRotateFlag = 1 Then
      imgCurrentMoon.Picture = _
              imgMoon(gCurrentMoon).Picture
      gCurrentMoon = gCurrentMoon + 1
        If gCurrentMoon = 8 Then
        gCurrentMoon = 0
    End If

    gLeftCorner = gLeftCorner + 10 * gDirection
    gTopCorner = gTopCorner + 10 * gDirection
    gWidthOfMoon = gWidthOfMoon + 10 * gEnlargeShrink
    gHeightOfMoon = gHeightOfMoon + 10 * gEnlargeShrink

    If gWidthOfMoon > 700 Then
       gEnlargeShrink = -1
```

```
    End If

    If gWidthOfMoon < 10 Then
       gEnlargeShrink = 1
    End If

    If imgCurrentMoon.Top > frmMoon.ScaleHeight Then
      gDirection = -1
    End If

    If imgCurrentMoon.Top < 10 Then
      gDirection = 1
    End If

End If
```

- احفظ المشروع باختيار **Save Project** من القائمة **File** لفيجول بيسك.

تنفيذ النسخة المحسنة لبرنامج القمر

دعنا نشاهد النسخة المحسنة لبرنامج القمر على أرض الواقع:

- نفّذ برنامج القمر.

وكما تشاهد يدور القمر حول محوره ويظهر أيضاً وكأنه يتحرك في ثلاثة أبعاد.

نص قسم التصاريح العامة

يحوي قسم التصاريح العامة للنموذج frmMoon على تصاريح إضافية عن المتحولات التي يفترض أن تكون مرئية لكل إجراءات النموذج.

نص الإجراء Form_Load()

يسند هذا الإجراء قيماً ابتدائية إلى المتحولات. فالمتحولان gLeftCorner و gTopCorner يمثلان الموقع الابتدائي للزاوية اليسرى العليا لعنصر التحكم imgCurrentMoon:

```
gLeftCorner = imgCurrentMoon.Left
gTopCorner = imgCurrentMoon.Top
```

كما أن الخاصية Stretch لعنصر التحكم imgCurrentMoon تساوي True

```
imgCurrentMoon.Stretch = True
```

مما يعني أن حجم العنصر imgCurrentMoon سوف يتغير بحيث يتسع ضمن عنصر التحكم.

هذه الخطوة ضرورية لأن حجم الصورة imgCurrentMoon سوف يزاد وينقص أثناء تنفيذ برنامج القمر.

نص الإجراء tmrTimer_Timer()

كتلة If الأولى في الإجراء tmrTimer_Timer() هي ذاتها في النسخة السابقة لهذا الإجراء، وتعتبر مسؤولة عن إظهار أحد صور مصفوفة التحكم.

تستخدم الطريقة Move حال ظهور الصورة Image لتحريكها، وهكذا فان الموقع الجديد للزاوية اليسرى العليا للصورة (صورة القمر)، يتوضع عند الإحداثيين X = gLeftCorner و Y = gTopCorner.

تستخدم طريقة Move الوسيطين الاختياريين Width (العرض) و Height (الارتفاع):

```
imgCurrentMoon.Moveg gLeftCorner , gTopCorner , _
                     gWidthOfMoon , gHeightOfMoon
```

وبعد التحريك، تمتلك الصورة عرضاً وارتفاعاً جديدين.

تجهز العبارات التالية لعبارة Move المتحولات التالية للتنفيذ التالي للإجراء tmrTimer_Timer():

```
gLeftCorner = gLeftCorner + 10 * gDirection
gTopCorner = gTopCorner + 10 * gDirection
gWidthOfMoon = gWidthOfMoon + 10 * gEnlargeShrink
gHeightOfMoon = gHeightOfMoon + 10 * gEnlargeShrink
```

ستزيد قيمة المتحولات السابقة أو تنقص، اعتماداً على قيمة المتحولات gDirection و gEnlargeShrink، فإذا كانت قيمة gDirection تساوي "1"، تزاد قيمة gLeftCorner كمثال بمقدار 10 Twips، أما إذا كانت قيمة gDirection تساوي "-1" فسوف تنقص قيمة gLeftCorner بمقدار 10 Twips).

يلي ذلك كله تفحص قيمة gWidthOfMoon، هل هي كبيرة جداً أم صغيرة جداً ؟!:

```
If gWidthOfMoon > 700 Then
```

```
   gEnlargeShrink = -1
End If

If gWidthOfMoon < 10 Then
   gEnlargeShrink = 1
End If
```

يختبر نص الإجراء بعد ذلك الموقع الحالي للزاوية اليسرى العليا من الصورة Image للتأكد من عدم تقاطعهما مع أسفل أو أعلى النموذج:

```
If imgCurrentMoon.Top > frmMoon.ScaleHeight Then
  gDirection = -1
End If

If imgCurrentMoon.Top < 10 Then
  gDirection = 1
End If
```

الخلاصة

تعلمنا في هذا الفصل كيفية استخدام عناصر التحكم الرسومية، كما تعلمنا كيفية إظهار ملفات الصور في عنصري التحكم Image و Picture وكيفية تحريك هذه الصور. ولا تنس أن مصفوفة عناصر التحكم تقدم تقنية مفيدة، تُستخدم غالباً في التحريك Animation.

الفصل الثامن

أساليب الرسم المباشرة

يركز هذا الفصل على أساليب رسومية أخرى، تتشابه مع عناصر التحكم الرسومية التي تناولناها في الفصل السابع. يعتبر استخدام الأساليب الرسومية الأخرى، أسهل في بعض الأحيان من استخدام عناصر التحكم الرسومية.

برنامج النقاط

يرسم برنامج النقاط نقاطاً في مواقع عشوائية من النموذج، موضحاً كيف تستخدم الطريقة Pset لرسم النقاط.

التمثيل المرئي لبرنامج النقاط

سنبدأ كعادتنا عند تصميم البرنامج بتمثيل نموذج البرنامج:

- أنشئ الدليل C:\VB5Prg\Ch08، لأننا سنحفظ عملنا في هذا الدليل.
- أنشئ مشروعاً تنفيذياً قياسياً جديداً Standard EXE، واحفظ نموذج المشروع بالاسم Points.frm في الدليل C:\VB5Prg\Ch08، واحفظ المشروع بالاسم Points.vbp في ذات الدليل.
- أنشئ النموذج frmPoints طبقاً للجدول 8-1.

يُفترض أن يبدو النموذج المكتمل كذلك المبين في الشكل 8-1.



الشكل 8-1

النموذج frmPoints

(طور التصميم).

الجدول 8-1. جدول خواص النموذج frmPoints.

القيمة	الخاصية	الكائن

Form	**Name**	**frmPoints**
	Caption	برنامج النقاط
	RightToLeft	True
Timer	**Name**	**tmrTimer**
	Interval	60
	Enabled	True
Menu	(انظر الجدول 8−2)	(انظر الجدول 8−2)

جدول 8−2. جدول قوائم النموذج frmPoints.

العنوان	الخاصية Name
&ملف	mnuFile
&خروج...	mnuExit
&رسوم	mnuGraphics
ار&سم النقاط...	mnuDrawPoints
ا&مسح النقاط...	mnuClear

إدخال نص برنامج النقاط

سنكتب الآن نص برنامج النقاط:

- أدخل النص التالي ضمن قسم التصاريح العامة للنموذج frmPoints:

```
'يجب التصريح عن كل المتحولات
Option Explicit
'متحول لمعرفة هل سوف ترسم النقاط أم سوف تمسح
Dim gDrawPoints
```

❑ أدخل النص التالي ضمن الإجراء Form_Load() للنموذج frmPoints:

```
Private Sub Form_Load()
    'تعطيل الرسم
    gDrawPoints = 0
End Sub
```

❑ أدخل النص التالي ضمن الإجراء mnuClear_Click():

```
Private Sub mnuClear_Click()
     'تعطيل الرسم
    gDrawPoints = 0
    'تنظيف سطح النموذج
    frmPoints.Cls
End Sub
```

❑ أدخل النص التالي ضمن الإجراء mnuDrawPoints_Click() للنموذج frmPoints:

```
Private Sub mnuDrawPoints_Click()
    'تفعيل الرسم
    gDrawPoints = 1
End Sub
```

❑ أدخل النص التالي ضمن الإجراء mnuExit_Click للنموذج frmPoints:

```
Private sub mnuExit_Click()

    End
End sub
```

❑ أدخل النص التالي ضمن الإجراء tmrTimer1_Timer() للنموذج frmPoints:

```
Private Sub tmrTimer_Timer()
    Dim R, G, B
    Dim X, Y
    Dim Counter

    'إذا كان الرسم مفعلا نفّذ ما يلي
    If gDrawPoints = 1 Then
      'ارسم 100 نقطة عشوائية
      For Counter = 1 To 100 Step 1
        'الحصول على لون عشوائي
        R = Rnd * 255
```

```
      G = Rnd * 255
      B = Rnd * 255
      'الحصول على موقع احداثيات عشوائي
      X = Rnd * frmPoints.ScaleWidth
      Y = Rnd * frmPoints.ScaleHeight
      'ارسم النقطة
      frmPoints.PSet (X, Y), RGB(R, G, B)
    Next
  End If
End Sub
```

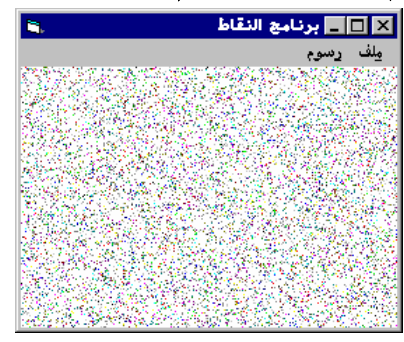
- احفظ المشروع باختيار **Save Project** من القائمة **File** لفيجول بيسك.

تنفيذ برنامج النقاط

دعنا نشاهد برنامج النقاط قيد التنفيذ:

- نفّذ برنامج النقاط.
- اختر **ارسم النقاط** من القائمة **رسوم**.

يستجيب البرنامج بإظهار نقاط ذات ألوان عشوائية في مواقع عشوائية من النموذج (انظر الشكل 8-2).



الشكل 8-2

رسم نقاط عشوائية

على سطح برنامج النقاط.

- اختر **امسح النقاط** من القائمة **رسوم**.

يستجيب البرنامج بمسح سطح برنامج النقاط.

- اختر **خروج** من القائمة **ملف** لإنهاء البرنامج.

كيف يعمل برنامج النقاط

يستعمل برنامج النقاط، الطريقة الرسومية Pset لرسم نقطة ما في النموذج، كما يستخدم الطريقة Cls لمحو النموذج.

نص قسم التصاريح العامة

يُصرَح عن المتحول gDrawPoints في قسم التصاريح العامة، ويخدم هذا المتحول كراية Flag. فإذا كان مساوياً إلى الواحد (أي الراية مرفوعة)، فهذا يعني تمكين عملية الرسم، وإذا ساوى الصفر، فهذا يعني عدم تمكين الرسم (الراية غير مرفوعة).

نص الإجراء Form_Load

يُنفّذ هذا الإجراء آلياً عند بدء تشغيل البرنامج:

```
Private Sub Form_Load()
    'تعطيل الرسم
    gDrawPoints = 0
End Sub
```

يسند هذا الإجراء القيمة الابتدائية صفر إلى المتحول gDrawPoints لمنع أو لعدم تمكين رسم النقاط.

نص الإجراء mnuClear_Click()

ينفذ هذا الإجراء عند اختيار **امسح النقاط** من القائمة **رسوم:**

```
Private Sub mnuClear_Click()
     'تعطيل الرسم
    gDrawPoints = 0
    'تنظيف سطح النموذج
    frmPoints.Cls
End Sub
```

يمنع هذا الإجراء عملية الرسم بإسناد القيمة صفر، إلى المتحول gDrawPoints ثم يستخدم الطريقة Cls لتنظيف سطح البرنامج من الرسوم التي عليه.

ملاحظة

استخدم الطريقة Cls (**Cl**ean **S**creen) لتنظيف محتويات سطح كائن ما، تمسح هذه الطريقة الرسوم المتولدة أثناء مرحلة التنفيذ بواسطة طرق الرسم المختلفة.

لاحظ مثلاً أن استخدام العبارة frmMyForm.Cls يؤدي إلى تنظيف سطح النموذج frmMyForm من أي رسوم متولدة عليه.

نص الإجراء mnuDrawPoints_Click()

ينفذ الإجراء mnuDrawPoints_Click() عند اختيار **ارسم النقاط** من القائمة **رسوم**:

```
Private Sub mnuDrawPoints_Click()
    'تفعيل الرسم
    gDrawPoints = 1
End Sub
```

يسند هذا الإجراء القيمة واحد إلى الراية (المتحول) gDrawPoints لتمكين الرسم.

نص الإجراء tmrTimer_Timer()

ينفذ الإجراء tmrTimer_Timer() تلقائياً كل 60 ميلي ثانية، وذلك بسبب إسناد القيمة 60 إلى الخاصية Interval:

```
Private Sub tmrTimer_Timer()
    Dim R, G, B
    Dim X, Y
    Dim Counter

    'إذا كان الرسم مفعلا نفّذ ما يلي
    If gDrawPoints = 1 Then
      'ارسم 100 نقطة عشوائية
      For Counter = 1 To 100 Step 1
        'الحصول على لون عشوائي
        R = Rnd * 255
        G = Rnd * 255
        B = Rnd * 255
        'الحصول على موقع احداثيات عشوائي
        X = Rnd * frmPoints.ScaleWidth
```

```
      Y = Rnd * frmPoints.ScaleHeight
      'ارسم النقطة
      frmPoints.PSet (X, Y), RGB(R, G, B)
    Next
  End If
End Sub
```

تفحص العبارة if قيمة الراية gDrawPoints، فإذا كانت مساوية إلى "الواحد"، فهذا يعني أنك اخترت **ارسم النقاط** من القائمة **رسوم**، ولذلك تنفذ كتلة العبارة If.

ترسم الحلقة For مائة نقطة، وكل نقطة ترسم بلون عشوائي في موقع عشوائي.

تستخدم الطريقة الرسومية PSet لرسم كل نقطة من هذه النقاط:

```
frmPoints.PSet (X, Y), RGB(R, G, B)
```

طريقة الرسم PSet

حسب ما أوضحنا، ترسم الطريقة PSet نقطة عند الإحداثيين X و Y المحددين ضمنها

(بواسطة وسائطها).

تمتلك الطريقة PSet وسيطاً إضافياً يدعى Step، وعند استخدام الوسيط Step، تُرسم النقطة نسبة إلى الإحداثيين CurrentX (الإحداثي X الحالي) وCurrentY (الإحداثي Y الحالي).

فمثلاً، لنفترض أن الإحداثي X الحالي CurrentX يساوي 100، والإحداثي Y الحالي CurrentY يساوي 50.

وهكذا يؤدي استخدام العبارة التالية:

```
frmPoints.PSet Step (10,20)، RGB(R,G,B)
```

إلى رسم نقطة عند الموقع (X=110 و Y=70)، تحُدَّث قيمة المتحولين CurrentX وCurrentY مباشرة (أي تلقائياً) بعد تنفيذ هذه العبارة، وتصبح قيمة CurrentX = 110 و CurrentY = 70.

تذكر أننا اصطلحنا منذ الفصل الرابع بأن فيجول بيسك، يحدّث قيمة الإحداثيين CurrentX و CurrentY تلقائياً، بإسناد إحداثيي نقطة النهاية لآخر رسم إليهما).

◻استبدل نص الإجراء tmrTimer_Timer() بالنص التالي:

```
Private Sub tmrTimer1_Timer()
    Dim R, G, B
    Dim X, Y
    Dim Counter

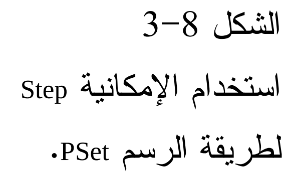
    If gDrawPoints = 1 Then
      For Counter = 1 To 100 Step 1
        R = Rnd * 255
        G = Rnd * 255
        B = Rnd * 255
        frmPoints.PSet Step(1, 1), RGB(R, G, B)
        If CurrentX >= frmPoints.ScaleWidth Then
           CurrentX = Rnd * frmPoints.ScaleWidth
        If CurrentY >= frmPoints.ScaleHieght Then
           CurrentY = Rnd * frmPoints.ScaleHieght
        End If
      Next
    End If
End Sub
```

◻احفظ المشروع بالطريقة المعتادة.

◻نفّذ البرنامج، واختر **ارسم النقاط** من القائمة **رسوم**، واترك البرنامج يعمل لفترة.

يرسم البرنامج الآن خطوطاً، حسب ما يبينه الشكل 8-3.




الشكل 8-3

استخدام الإمكانية Step

لطريقة الرسم PSet.

يستخدم نص الإجراء tmrTimer1_Timer() الطريقة PSet مع الإمكانية Step:

```
frmPoints.PSet Step (1,1)، RGB(R,G,B)
```

تبعد كل نقطة مرسومة عن سابقتها بمقدار وحدة إلى اليمين ووحدة إلى الأسفل. وهذا يشرح سبب رسم النقاط وكأنها خطوط مستقيمة حسب ما يوضحه الشكل 8-3.

تتحقق عبارتي If في هذا الإجراء، من أن النقاط المرسومة لا تتجاوز حدود النموذج.

ملاحظة

استخدم الطريقة PSet (**Point Set**) لرسم نقطة في كائن ما، وذلك كالتالي:

ObjectName.Pset (X,Y),Color

تضع العبارة السابقة نقطة على الكائن عند الموقع (X,Y) وبلون معين.

لوضع نقطة معينة على الكائن نسبة إلى آخر نقطة تم رسمها، استخدم العبارة التالية:

ObjectName.Pset Step (X,Y),Color

وهي مكافئة للعبارة التالية:

ObjectName.Pset (oldX + X,oldY + Y),Color

حيث oldX و oldY، إحداثيات آخر نقطة تم رسمها على الكائن.

الطريقة Point

تعود هذه الطريقة بقيمة لون نقطة ضوئية Pixel محددة. فمثلاً، لمعرفة لون النقطة الضوئية المحددة بالموقع 40 و 30، استخدم العبارة التالية:

```
PixelColor = Point (30,40)
```

ورغم أن برنامج النقاط لم يستخدم الطريقة Point، إلا أنك قد تجد استخداماً لها في مشاريعك المستقبلية.

رسم الخطوط Line

تسمح الطريقة Line للمستخدم برسم خطوط، وتمتلك الصيغة التالية:

```
Line(X1,Y1)-(X2,Y2), Color
```

علماً أن (X1,Y1) يمثل إحداثي نقطة بداية الخط، ويمثل (X2,Y2) إحداثي نقطة نهاية الخط، كما أن Color يمثل لون الخط. ويؤدي تجاهل كتابة النقطة الأولى (X1,Y1) إلى رسم الخط بدءاً من الإحداثيين CurrentX و CurrentY. وذلك كالتالي:

```
Line -(X2,Y2), Color
```

سنضيف عنصر قائمة جديد يدعى **رسم خطوط** إلى نظام قوائم برنامج النقاط، بغية توضيح عمل طريقة Line، يعطي الجدول 8-3 جدول القوائم الجديد لبرنامج النقاط.

الجدول 8-3. جدول القوائم الجديد للنموذج frmPoints.

العنوان	الاسم
&ملف	mnuFile
...&خروج	mnuExit
&رسوم	mnuGraphics
...ار&سم النقاط	mnuDrawPoints
...ا&مسح النقاط	mnuClear
...رسم &خطوط	mnuLines

❑ أضف النص التالي إلى الإجراء ()mnuLines_Click للنموذج frmPoints:

```
Private sub mnuLines_Click()
    Line -(Rnd * frmPoints.ScaleWidth, _
           Rnd * frmPoints.ScaleHeight),RGB(0,0,0)
End sub
```

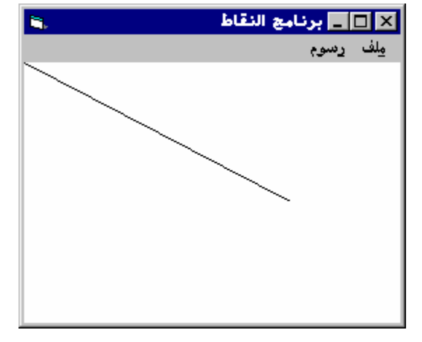
يرسم هذا الإجراء خطاً واحداً من آخر نقطة رسم وصل إليها، إلى نقطة عشوائية.

❑ احفظ المشروع بالطريقة المعتادة.

❑ نفّذ البرنامج النقاط.

❑ اختر **رسم خطوط** من القائمة **رسوم**.

يُرسم خط في النموذج، ويتشابه مع ذاك المبين في الشكل 8-4.



الشكل 8-4

رسم خط ما

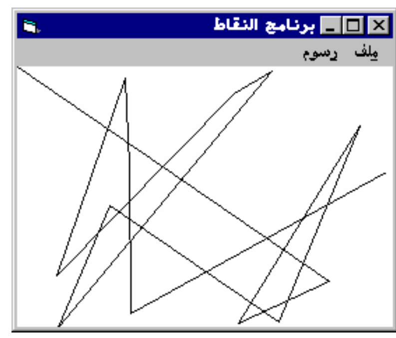
بواسطة الطريقة Line.

بسبب تجاهل الإحداثيين (X1,Y1)، ترسم عبارة Line في هذا الإجراء خطاً ما، ابتداءً من الإحداثيين CurrentX و CurrentY، تكون القيمة الابتدائية للإحداثيين CurrentX و CurrentY مساوية إلى الصفر عند بدء تشغيل البرنامج. وهذا يشرح سبب بداية الخط في الشكل 8-4 من الإحداثيين 0,0.

- اختر **رسم خطوط** من القائمة **رسوم** عدة مرات.

يُرسم خط جديد في كل مرة، ويبدأ من نهاية الخط السابق (انظر الشكل 8-5).

- اختر **خروج** من القائمة **ملف** لإنهاء البرنامج.



الشكل 8-5

رسم خطوط متصلة.

دعنا الآن نكتب برنامجاً يرسم عدة خطوط عند اختيار **رسم خطوط** من القائمة **رسوم**:

```
Private Sub mnuLines_Click()
   Dim Counter
   For Counter = 1 To 100 Step 1
     Line -(Rnd * frmPoints.ScaleWidth, _
            Rnd * frmPoints.ScaleHeight), RGB(0, 0, 0)
   Next
End Sub
```

- نفّذ البرنامج.
- اختر **رسم خطوط** من القائمة **رسوم**.

يرسم البرنامج مائة خط متصل، حسب ما يبينه الشكل8-6.



الشكل 8-6

رسم مائة خط متصل.

استخدام الوسيط Step في الطريقة Line

يمكنك استخدام الوسيط Step الاختياري في الطريقة Line، كما يلي:

```
Line (X1,Y1) - Step (dx,dy),Color
```

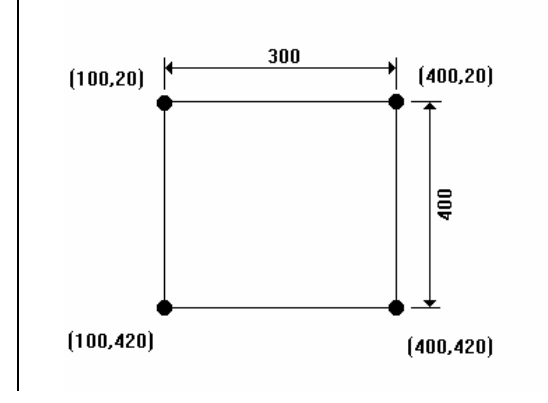
علماً أن (X1,Y1) تمثل إحداثيات نقطة البداية، أما Step (dx,dy) فهي دلالة لفيجول بيسك، بأن نقطة نهاية الخط تقع عند الإحداثيين X1 + dx و Y1 + dy. فمثلاً ترسم العبارة التالية:

```
Line (20,30) - Step (50,100), RGB(0,0,0)
```

خطاً يبدأ من الإحداثيين 30 , 20 وينتهي بالإحداثيين 130 , 70.

يمكن استخدام الإمكانية Step أيضاً لرسم مربع. ترسم العبارات التالية المربع المبين في الشكل 8-7:

```
Line (10,20) - (400,20)
Line -Step(0,400)
Line -Step(-300,0)
Line -Step(0,-400)
```



الشكل 8-7

رسم مربع بطريقة

رسم الخط Line.

كما تشاهد تعتبر عبارات Line الأربعة لازمة لرسم مربع واحد، الطريقة الأسهل لرسم مربع، بواسطة الطريقة Line تكون باستخدام Line مع الخيار B، أي:

```
Line (100,20)-(400,420),RGB(0,0,0),B
```

الإحداثيان الأولين هما إحداثيا الزاوية اليسرى العليا للمربع، أما الإحداثيان اللذان يليهما فيمثلان إحداثيي الزاوية السفلى اليمنى للمربع. يطلب الوسيط B من فيجول بيسك، رسم مربع له هاتين الزاويتين.

إذا أردت ملء المربع (طمسه)، استخدم الوسيط F:

```
Line(100,20)-(400,420),RGB(0,255,0),BF
```

ترسم هذه العبارة مربعاً وتملؤه باللون الأخضر، لا توجد فاصلة بين B و F لأنك لا تستطيع استخدام F بدون B.

ملاحظة

يقصد برسم المربع هنا، رسم شكل رباعي الأضلاع، وهو ليس بالضرورة مربعاً متساوي الأضلاع.

ملء المربع بواسطة الخاصية FillStyle

تُقدم الخاصية FillStyle طريقة ثانية لملء المربع المرسوم، فباستخدام الخاصيتين FillColor و FillStyle للنموذج، واستخدام طريقة Line مع الوسيط B دون الوسيط F نستطيع ملء المربع:

```
frmMyForm.FillStyle = 2
frmMyForm.FillColor = RGB(255,0,0)
frmMyForm.Line (100,20)-(400,420),RGB(0,0,0),B
```

تسند العبارة الأولى القيمة 2 إلى الخاصية FillStyle. يبين الجدول 8–4 معاني القيم النهائية المحتملة لهذه الخاصية، وعندما تكون قيمة FillStyle مساوية إلى 2، يمُلأ الكائن بخطوط أفقية.

تسند العبارة الثانية اللون الأحمر RGB(255,0,0) إلى الخاصية FillColor، (بمعنى أن المربع سيلون بالأحمر)، وهكذا تتسبب العبارات الثلاث برسم مربع وملؤه بخطوط شاقولية حمراء.

الجدول 8-4. القيم الثمانية المحتملة للخاصية FillStyle.

القيمة	الشرح
0	لون خالص مصمت Solid.
1	لون شفاف غير مرئي Transparent (القيمة الافتراضية).
2	خطوط أفقية Horizontal Lines.
3	خطوط عمودية Vertical Lines.
4	خطوط مائلة للأعلى Upward diagonal Lines.
5	خطوط مائلة للأسفل Downward diagonal Lines.
6	خطوط متقاطعة Cross hatch
7	خطوط متقاطعة مائلة Diagonal Cross hatch.

للتعرف على معاني كل قيمة من قيم الخاصية FillStyle في الجدول 8-4 قم بما يلي:

- أضف القائمة **رسم مربع** إلى نظام قوائم النموذج frmPoints.

يعطي الجدول 8-5 جدول القائمة الجديد.

الجدول 8-5. إضافة القائمة رسم مربع إلى النموذج frmPoints.

العنوان	الخاصية Name
&ملف	mnuFile
&خروج...	mnuExit
&رسوم	mnuGraphics
ار&سم النقاط...	mnuDrawPoints

...ا&مسح النقاط	mnuClear
...رسم &خطوط	mnuLines
رسم مربع	mnuDrawBox
...أحمر	mnuRed
...أخضر	mnuGreen
...أزرق	mnuBlue
...−	mnuSep1
...تغيير نمط الرسم	mnuSetStyle

❑ أضف النص التالي ضمن الإجراء ()mnuRed_Click للنموذج frmPoints:

```
Private Sub mnuRed_Click()
    'تغيير الخاصية FillColor إلى الأحمر
    frmPoints.FillColor = RGB(255, 0, 0)
    'رسم المربع
    frmPoints.Line (100, 80)-Step(5000, 3000), _
                RGB(255, 0, 0), B
End Sub
```

❑ أضف النص التالي ضمن الإجراء ()mnuBlue_Click للنموذج frmPoints:

```
Private Sub mnuBlue_Click()
    'تغيير الخاصية FillColor إلى الأحمر
    frmPoints.FillColor = RGB(255, 0, 0)
    'رسم المربع
    frmPoints.Line (100, 80)-Step(5000, 3000), _
                 RGB(0, 0, 255), B
End Sub
```

❑ أضف النص التالي ضمن الإجراء ()mnuGreen_Click للنموذج frmPoints:

```
Private Sub mnuGreen_Click()
    'تغيير الخاصية FillColor إلى الأحمر
    frmPoints.FillColor = RGB(255, 0, 0)
    'رسم المربع
    frmPoints.Line (100, 80)-Step(5000, 3000), _
                 RGB(0, 255, 0), B
End Sub
```

❑أضف النص التالي ضمن الإجراء ()mnuSetStyle_Click للنموذج

:frmPoints

```
Private Sub mnuSetStyle_Click()
    Dim FromUser
    Dim Instruction
    Instruction = "أدخل رقماً (0-7) لتعيينه للخاصية FillStyle"
    'الحصول على قيمة من المستخدم
    FromUser = InputBox$(Instruction, _
                         "تعيين قيمة FillStyle ")
    'تنظيف النموذج
    frmPoints.Cls
    'التأكد من القيمة المدخلة
    If Val(FromUser) >= 0 And Val(FromUser) <= 7 Then
      frmPoints.FillStyle = Val(FromUser)
    Else
      Beep
      MsgBox ("قيمة غير صحيحة")
    End If
    'رسم المربع
    frmPoints.Line (100, 80)-Step(5000, 3000), _
                  RGB(0, 0, 0), B
End Sub
```

❑ احفظ المشروع، باختيار **Save Project** من القائمة **File** لفيجول بيسك.

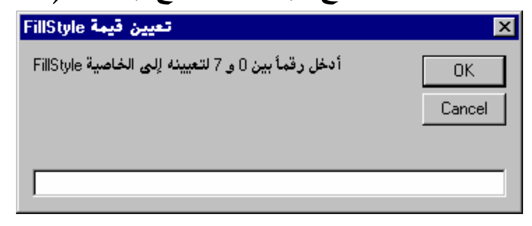
تنفيذ برنامج النقاط

لنشاهد ما كتبناه قيد العمل:

❑نفّذ برنامج النقاط.

❑اختر **تغيير نمط الرسم** من القائمة **رسم مربع**.

يستجيب البرنامج بإظهار مربع إدخال (Input Box) حسب ما يظهر من الشكل 8-8.




الشكل 8-8

إدخال قيمة

للخاصية FillStyle.

❑أدخل رقماً بين صفر و سبعة، وانقر الزر **OK**. يمثل هذا الرقم قيمة الخاصية FillStyle.

يُظهر البرنامج المربع ويملؤه حسب قيمة FillStyle المدخلة في الخطوة السابقة. يظهر في الشكل 8–9 المربع عندما تكون قيمة FillStyle مساوية إلى 4 (خطوط مائلة). قد تضطر إلى سحب الحافة اليسارية للإطار لتوسيعه.



الشكل 8–9 التمرن على الخاصية FillStyle.

❑ اختر **أحمر** أو **أزرق** أو **أخضر** من القائمة **رسم مربع**.

يستجيب البرنامج برسم المربع باللون المنتقى.

❑ تمرن على برنامج النقاط، واختر **خروج** من القائمة **ملف** لإنهائه.

نص الإجراء mnuRed_Click()

ينفذ الإجراء mnuRed_Click عند اختيار **أحمر** من القائمة **رسم مربع**:

```
Private Sub mnuRed_Click()
    'تغيير الخاصية FillColor إلى الأحمر
    frmPoints.FillColor = RGB(255, 0, 0)
    'رسم المربع
    frmPoints.Line (100, 80)-Step(5000, 3000), _
                RGB(255, 0, 0), B
End Sub
```

يحدد هذا الإجراء قيمة الخاصية FillColor باللون الأحمر، ثم يستخدم الطريقة Line مع الوسيط B لرسم مربع. يملأ المربع تبعاً للقيمة الراهنة للخاصية FillStyle للنموذج.

يعمل الإجراءان mnuBlue_Click() و mnuGreen_Click() بطريقة مشابهة، محددين قيمة الخاصية FillColor باللون الأزرق Blue أو الأخضر Green.

نص الإجراء mnuSetStyle_Click

ينفذ هذا الإجراء عند اختيار **تغيير نمط الرسم** من القائمة **رسم مربع:**

```
Private Sub mnuSetStyle_Click()
    Dim FromUser
    Dim Instruction
    Instruction = "FillStyle لتعيينه للخاصية (7-0) أدخل رقماً"
    'الحصول من المستخدم على القيمة
    FromUser = InputBox$(Instruction, _
          "FillStyle تعيين قيمة ")
    'تنظيف النموذج
    frmPoints.Cls
    'التأكد من القيمة المدخلة
    If Val(FromUser) >= 0 And Val(FromUser) <= 7 Then
    frmPoints.FillStyle = Val(FromUser)
    Else
     Beep
     MsgBox ("قيمة غير صحيحة")
    End If
    'رسم المربع
    frmPoints.Line (100, 80)-Step(5000, 3000), _
                  RGB(0, 0, 0), B
End Sub
```

يستخدم هذا الإجراء التابع الوظيفي InputBox() للحصول على رقم بين صفر وسبعة، ثم ينظف النموذج بواسطة الطريقة Cls، يلي ذلك التحقق من القيمة التي أدخلها المستخدم وذلك باستخدام العبارة If بغية التحقق من أن القيمة تقع ضمن المجال الصحيح (المسموح به).

فإذا كانت القيمة المدخلة واقعة بين صفر وسبعة،تُحَدَّث عند ذلك الخاصية FillStyle بإسناد تلك القيمة إليها.

ترسم آخر عبارة في الإجراء مربعاً باستخدام الطريقة Line مع الإمكانية B. يُرسم المربع اعتماداً على القيمتين الحاليتين للخاصتين FillColor و FillStyle للنموذج.

برنامج الدائرة

نُقدم الآن طريقة رسم هامة أخرى وهي الطريقة Circle، تُستخدم هذه الطريقة لرسم الدوائر والقطوع الناقصة، ويوضح برنامج الدائرة كيفية رسم الدوائر.

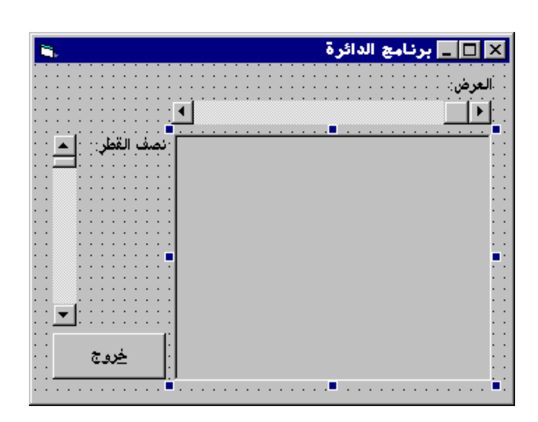
يمكنك استخدام الطريقة Circle لرسم الدوائر سواء في النموذج، أوفي عنصر تحكم الصورة.

التمثيل المرئي لبرنامج الدائرة

نبدأ كعادتنا بطور التمثيل المرئي (طور التصميم) لنموذج برنامج الدائرة:

- أنشئ مشروعاً تنفيذياً قياسياً جديداً Standard EXE، واحفظ نموذج المشروع بالاسم Circles.frm في الدليل C:\VB5Prg\Ch08، واحفظ المشروع بالاسم Circles.vbp في الدليل C:\VB5Prg\Ch08.
- أنشئ النموذج frmCircles وفقاً للجدول 8-6.

يفترض أن يبدو النموذج المكتمل كما في الشكل 8-10.



الشكل 8-10

النموذج frmCircles

في طور التصميم.

الجدول 8-6 جدول خصائص النموذج frmCircles.

الكائن	الخاصية	القيمة
Form	**Name**	**frmCircles**
	Caption	برنامج الدائرة
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
Picture Box	**Name**	**picCircles**
	BorderStyle	1-Fixed Single
Vertical Scroll Bar	**Name**	**vsbRadius**
	Max	100
	Min	1
	Value	1
Horizontal Scroll Bar	**Name**	**hsbCircleWidth**
	Max	10
	Min	1
	Value	1
Label	**Name**	**lblWidth**
	Caption	العرض:
	RightToLeft	True
Label	**Name**	**lblRadius**
	Caption	نصف القطر:
	RightToLeft	True

إدخال نص برنامج الدائرة

سندخل الآن نص برنامج الدائرة:

- تحقق من احتواء قسم التصاريح العامة على العبارة Option Explicit، أي:

```
' يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء cmdExit_Click() للنموذج frmCircles:

```
Private Sub cmdExit_Click()
```

```
    End
End Sub
```

- أدخل النص التالي ضمن الإجراء ()hsbCircleWidth_Change:

```
Private Sub hsbCircleWidth_Change()
    Dim X, Y, Radius
    picCircles.DrawWidth = hsbCircleWidth.Value
    X = picCircles.ScaleWidth / 2
    Y = picCircles.ScaleHeight / 2
    picCircles.Cls
    picCircles.Circle (X, Y), vsbRadius.Value * 10, _
              RGB(255, 0, 0)
End Sub
```

- أدخل النص التالي ضمن الإجراء ()hsbCircleWidth_Scroll:

```
Private Sub hsbCircleWidth_Scroll()
    hsbCircleWidth_Change
End Sub
```

- أدخل النص التالي ضمن الإجراء ()vsbRadius_Change:

```
Private Sub vsbRadius_Change()
    Dim X, Y, Radius
    X = picCircles.ScaleWidth / 2
    Y = picCircles.ScaleHeight / 2
    picCircles.Cls
    picCircles.Circle (X, Y), vsbRadius.Value * 10, _
              RGB(255, 0, 0)
End Sub
```

- أدخل النص التالي ضمن الإجراء ()vsbRadius_Scroll:

```
Private Sub vsbRadius_Scroll()
    vsbRadius_Change
End Sub
```

- احفظ المشروع بالطريقة المعتادة.

تنفيذ برنامج الدائرة

لنشاهد ما كتبناه قيد التنفيذ:

- نفّذ برنامج الدائرة.

❑ غيِّر موقع مؤشر شريط التمرير العمودي.

يتغير نصف قطر الدائرة الظاهرة، وفقاً لوضعية مؤشر شريط التمرير العمودي (انظر الشكل 8-11).

❑غيِّر موقع مؤشر شريط التمرير الأفقي لتحديد قيمة جديدة لعرض حدود الدائرة. وترسم الدائرة بخط مختلف العرض.



الشكل 8-11

تغيير نصف قطر الدائرة

بواسطة شريط التمرير العمودي.

كيف يعمل برنامج الدائرة

يستخدم برنامج الدائرة الطريقة Circle لرسم الدائرة. يتغير نصف قطر الدائرة تبعاً لموقع مؤشر شريط التمرير العمودي، ويتغير عرض حدود الدائرة تبعاً لموقع مؤشر شريط التمرير الأفقي.

نص الإجراء vsbRadius_Change()

ينفذ الإجراء vsbRadius_Change() عند تغيير موقع مؤشر شريط التمرير الأفقي:

```
Private Sub vsbRadius_Change()
    Dim X, Y, Radius
    X = picCircles.ScaleWidth / 2
    Y = picCircles.ScaleHeight / 2
    picCircles.Cls
    picCircles.Circle (X, Y), vsbRadius.Value * 10, _
                 RGB(255, 0, 0)
End Sub
```

يحسب هذا الإجراء، إحداثيي نقطة مركز الدائرة (x,y) بحساب إحداثيات مركز عنصر تحكم الصورة، بحيث يكون مركز الدائرة هو نفسه مركز عنصر تحكم الصورة.

يلي ذلك تنظيف المحتويات الرسومية السابقة الموجودة على عنصر تحكم الصورة، وذلك باستخدام الطريقة Cls (أي تنظيف العنصر من الدائرة السابقة في حال وجودها)، وأخيراً ترسم الدائرة باستخدام الطريقة Circle:

```
picCircles.Circle (X, Y), vsbRadius.Value * 10, _
                         RGB(255, 0, 0)
```

يكون نصف قطر الدائرة المرسومة بلون أحمر، أكبر عشر مرات من القيمة (الحالية) الراهنة لموقع مؤشر شريط التمرير العمودي.

ملاحظه

استخدم الطريقة Circle لرسم دائرة ما على الكائن:

ObjectName.Circle(X , Y) , Radius , color

فمثلاً، لرسم دائرة زرقاء على النموذج المدعو frmMyForm باحداثيي مركز 1000,500 ونصف قطر 75، استخدم العبارة التالية:

frmMyForm.Circle (1000 , 500), 75 , RGB(0,0,255)

بطريقة مشابهة، يرسم الإجراء vsbRadius_Scroll() دائرة جديدة عند تغيير موقع مؤشر شريط التمرير العمودي، وذلك باستدعاء الإجراء vsbRadius_Change.

نص الإجراء hsbCircleWidth_Change()

ينفذ الإجراء hsbCircleWidth_Change() عند تغيير موقع مؤشر شريط التمرير الأفقي:

```
Private Sub hsbCircleWidth_Change()
    Dim X, Y, Radius
    picCircles.DrawWidth = hsbCircleWidth.Value
    X = picCircles.ScaleWidth / 2
    Y = picCircles.ScaleHeight / 2
    picCircles.Cls
    picCircles.Circle (X, Y), vsbRadius.Value * 10, _
              RGB(255, 0, 0)
End Sub
```

يغيّر هذا الإجراء الخاصية DrawWidth لعنصر تحكم الصورة المدعو picCircles، تبعاً لموقع مؤشر شريط التمرير الأفقي. تحددِ قيمة الخاصية DrawWidth عرض الدائرة

المرسومة في عنصر تحكم الصورة. وكما تشاهد هناك تشابه كبير بين هذا الإجراء والإجراء vsbRadius_Change(). ولكن بدلاً من تغيير نصف القطر، يتم هنا تحديث الخاصية DrawWidth لعنصر تحكم الصورة:

```
picCircles.DrawWidth = hsbCircleWidth.Value
```

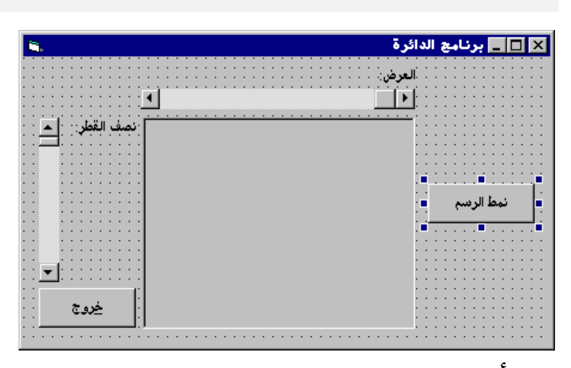
وبطريقة مشابهة، يرسم الإجراء hsbCircleWidth_Scroll() دائرة جديدة، عند تغيير عرض حدود الدائرة، عند تحريك مؤشر شريط التمرير الأفقي، وذلك باستدعاء الإجراء hsbCircleWidth_Change().

تحسين برنامج الدائرة

نستطيع بإضافة بعض الأسطر إلى برنامج الدائرة، الحصول على مؤثرات رسومية جذابة:

- أضف الزر **نمط الرسم** إلى النموذج frmCircles، حسب مايبينه الشكل 8-12.
- أسند لهذا الزر الخصائص التالية:

```
Name:          cmdDrawStyle
Caption:       نمط الرسم
RightToLeft:   True
```



الشكل 8-12

إضافة الزر **نمط الرسم**

إلى برنامج الدائرة.

- أضف النص التالي ضمن الإجراء cmdDrawStyle_Click() للنموذج frmCircles:

```
Private Sub cmdDrawStyle_Click()
    Dim TheStyle
    TheStyle = InputBox$("أدخل رقماً (6-0):")
    If Val(TheStyle) < 0 Or Val(TheStyle) > 6 Then
      Beep
      MsgBox ("إدخال غير صحيح")
```

```
    Else
      picCircles.DrawStyle = Val(TheStyle)
    End If
End Sub
```

❑احذف شريط التمرير الأفقي. (تأكد من اختيار شريط التمرير الأفقي ثم اضغط المفتاح Delete لحذفه).

كتبنا في النسخة السابقة لبرنامج الدائرة، نص الإجراء ()hsbWidth_Change، أما الآن، فقد حذفنا عنصر التحكم هذا، لهذا يضع فيجول بيسك نص هذا الإجراء في المنطقة General للنموذج frmCircles، لهذا تستطيع بعد حذفك عنصر التحكم، إزالة إجراءاته (Procedures) أيضاً من المنطقة General.

❑احذف الإجراء ()hsbWidth_Change من المنطقة General، وذلك بإضاءته كاملاً بما في ذلك أول وآخر سطرين فيه، ثم اضغط الزر Delete من لوحة المفاتيح.

❑احذف الإجراء ()hsbWidth_Scroll من المنطقة General بنفس الطريقة المتبعة في الخطوة السابقة.

❑ احذف اللافتة lblWidth باختيارها، ثم ضغط المفتاح Delete من لوحة المفاتيح.

❑حدد اللون الأبيض White في الخاصية BackColor لعنصر التحكم picCircles، مما يعني أن الدائرة سترسم على خلفية بيضاء، الأمر الذي يمكِّنك من مشاهدة تأثيرات نماذج الرسم بشكل أفضل.

تنفيذ برنامج الدائرة

دعنا نشاهد النص الذي كتبناه قيد التنفيذ:

❑ نفّذ برنامج الدائرة.

❑ انقر الزر **نمط الرسم**.

يستجيب البرنامج بإظهار مربع رسالة، يطلب منك إدخال رقم بين الصفر والستة.

❑أدخل رقماً بين الصفر والستة (هذا الرقم يمثل نمط الرسم)، ثم انقر على الزر **OK**.

- حرك مؤشر شريط التمرير العمودي جيئة وذهاباً، وشاهد التأثيرات التي تتسبب بها قيم مختلفة لأنصاف الأقطار والأنماط المختلفة، على الدائرة.
- أنهِ البرنامج بالنقر على الزر **خروج**.

الخاصية DrawStyle

يُنفذ الإجراء cmdDrawStyle_Click() عند نقر الزر **نمط الرسم**.

يُطلب منك إدخال رقم بين الصفر والستة، ويُستخدم هذا الرقم كقيمة للخاصية DrawStyle لعنصر تحكم الصورة. يعرض الجدول 8–7 قائمة بالقيم السبعة المحتملة للخاصية DrawStyle ومعانيها.

الجدول 8–7. القيم السبعة المحتملة للخاصية DrawStyle.

القيمة	الشرح
0	مصمت Solid (القيمة الافتراضية).
1	شَرطة Dash (− − −).
2	نقطة Dot (. . .).
3	شَرطة نقطة Dash Dot (− . − .).
4	شَرطة نقطة نقطة Dash Dot Dot (− . . − . .).
5	غير مرئي Invisible.
6	مصمت داخلي Inside Solid.

ملاحظة

تغير الخاصية DrawStyle، طريقة رسم الخطوط على الكائن، كما رأينا في الجدول السابق.

لكن هناك شرط واحد لكي تظهر هذه التأثيرات وهو: أن يكون عرض حدود الخطوط مساو للواحد (DrawWidth = 1)، ولا يمكن ظهور هذه التأثيرات، في حال ما إذا كانت قيمة الخاصية DrawWidth أكبر من الواحد.

تحسين برنامج الدائرة مجدداً

سنجري الآن تعديلاً إضافياً على برنامج الدائرة:

- استبدل نص الإجراء vsbRadius_Change() بالنص التالي:

```
Private Sub vsbRadius_Change()
    Dim X, Y, Radius
    Static LastValue
    Dim R, G, B
    R = Rnd * 255
    G = Rnd * 255
    B = Rnd * 255
    X = picCircles.ScaleWidth / 2
    Y = picCircles.ScaleHeight / 2
    If LastValue > vsbRadius.Value Then
      picCircles.Cls
    End If
    picCircles.Circle (X, Y), vsbRadius.Value * 10, _
                  RGB(R, G, B)
    LastValue = vsbRadius.Value
End Sub
```

- احفظ المشروع باختيار **Save Project** من القائمة **File**.

تنفيذ النسخة المحسنة من برنامج الدائرة

سنشاهد الآن بعض المؤثرات الرسومية الجميلة:

- نفّذ برنامج الدائرة.
- انقر على الزر **نمط الرسم**.

يستجيب البرنامج بإظهار مربع الدخول Input Box الذي يطالبك بإدخال رقم بين الصفر والستة.

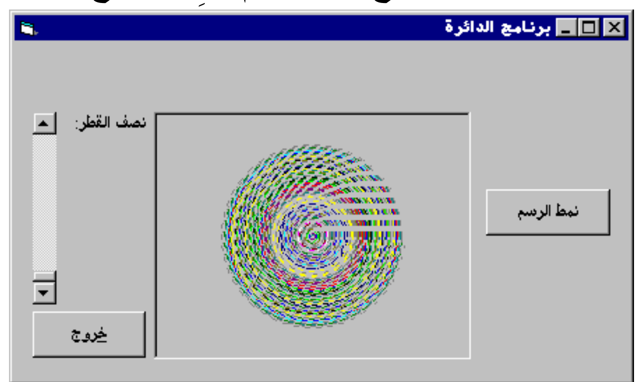
- أدخل الرقم 2 ثم انقر على الزر **OK**.
- زد موقع مؤشر شريط التمرير العمودي **نصف القطر** بالنقر عدة مرات، فوق رأس السهم النازل الواقع أسفل شريط التمرير هذا.

يستجيب البرنامج برسم دوائر كما في الشكل 8-13.

□أنقص موقع مؤشر شريط التمرير العمودي **نصف القطر** بالنقر عدة مرات، فوق رأس السهم الصاعد المتوضع أعلى شريط التمرير.

يستجيب البرنامج برسم دوائر متعاقبة أصغر.

□تمرن على برنامج الدائرة، ثم أنهِ البرنامج بنقر الزر **خروج**.



الشكل 8-13

رسم دوائر بواسطة النسخة المحسنة من برنامج الدائرة.

نص الإجراء vsbRadius_Change()

ينفذ الإجراء vsbRadius_Change() آلياً عند تغيير موقع مؤشر شريط التمرير.

يبدأ الإجراء بالتصريح عن بضعة متحولات، ويعرّف المتحول الثاني كمتحول ساكن Static:

```
Static LastValue
```

مما يعني أن المتحول LastValue لا تسند له القيمة الابتدائية صفر، كلما نُفذ هذا الإجراء، وبمعنى آخر، يحافظ المتحول على قيمته ضمن هذا الإجراء.

يجهز الإجراء بعد ذلك المتحولات الثلاثة R,G,B ويسند إليها أرقاماً تمثل ألواناً عشوائية:

```
'توليد لون عشوائي
R = Rnd * 255
G = Rnd * 255
B = Rnd * 255
```

يحسب الإجراء بعد ذلك إحداثيات مركز عنصر تحكم الصورة:

```
X = picCircles.ScaleWidth / 2
Y = picCircles.ScaleHeight / 2
```

تستخدم الإحداثيات بعد ذلك ضمن هذا الإجراء كإحداثيات لمركز الدوائر.

يحتفظ المتحول LastValue بقيمة الخاصية Value لشريط التمرير العمودي، قبل تغيير موقع مؤشر شريط التمرير.

تتحقق العبارة If إذا كان الموقع الحالي لمؤشر شريط التمرير (المحدد بالخاصية Value)، أصغر من آخر موقع لشريط التمرير:

```
If LastValue > vsbRadius.Value Then
  picCircles.Cls
End If
```

إذا كانت القيمة الراهنة للخاصية Value لشريط التمرير العمودي، أصغر من قيمة المتحول LastValue، فهذا يعني أنك أنقصت قيمة شريط التمرير. وفي هذه الحالة تنفذ الطريقة Cls التي تنظف عنصر تحكم لصورة.

يرسم الإجراء بعد ذلك دائرة باستخدام الطريقة Circle:

```
picCircles.Circle (X, Y), vsbRadius.Value * 10, RGB(R, G, B)
```

وآخر شيء يفعله هذا الإجراء، هو تحديث قيمة المتحول LastValue. وعند التنفيذ التالي لهذا الإجراء، تمثل قيمة المتحول LastValue الخاصية Value لشريط التمرير قبل التغيير:

```
LastValue = vsbRadius.Value
```

فإذا زاد المستخدم شريط التمرير، لا تنفذ الطريقة Cls، وهكذا تبقى الدوائر التي رسمت مسبقاً على الشاشة عند زيادة شريط التمرير، وينظف عنصر تحكم الصورة عند إنقاص شريط التمرير.

المتحول الساكن LastValue

لعل المتحول LastValue هو الشيء الهام الواجب الانتباه إليه، من المناقشة السابقة.

يصرح عن هذا المتحول بأنه Static:

```
Static LastValue
```

تُسند القيمة صفر إلى المتحول LastValue عند تنفيذ الإجراء vsbRadius_Change() لأول مرة من قبل فيجول بيسك. تتغير هذه القيمة أثناء تنفيذ الإجراء، وعند التنفيذ التالي

لهذا الإجراء، يُعاد إسناد القيمة الابتدائية صفر إلى المتحولات الغير ساكنة non static (مثل المتحول X و Y و Radius) من جديد.

أما المتحول LastValue فلا تتبدل قيمته منذ آخر تنفيذ للإجراء، وذلك بسبب التصريح عنه بأنه من النوع الساكن Static (بمعنى أنه يحتفظ بقيمته التي أسندت إليه من التنفيذ السابق للإجراء).

قد تلاحظ تشابهاً بين المتحول المصرح عنه في قسم التصاريح العامة General Declarations، وبين المتحول المصرح عنه كمتحول ساكن Static. ففي كلتا الحالتين، يحتفظ المتحول بقيمته طالما أن البرنامج يعمل. لكن هنالك اختلاف هام بين نوعي المتحولين هذين:

- المتحول المصرح عنه في قسم التصاريح العامة لنموذج يمكن الولوج إليه من ضمن أي إجراء في النموذج.
- المتحول المصرح عنه في إجراء كمتحول ساكن Static، يقبل الولوج إليه من ضمن الإجراء الذي صرّح عن هذا المتحول.

فإذا حاولت مثلاً الولوج إلى المتحول LastValue من ضمن الإجراء Form_Load() مثلاً، فسوف ينتج لديك خطأ، لأنك لا تستطيع الوصول إلى هذا المتحول من خارج الإجراء vsbRadius_Change()، يمكنك أيضاً التصريح عن متحول آخر يدعى LastValue ضمن الإجراء Form_Load(). وهذه القيمة يمكن التصريح عنها كمتحول ساكن أو متحول غير ساكن نظامي، لكن لا يوجد أي صلة بين المتحول LastValue المنتمي للإجراء Form_Load()، وبين المتحول LastValue الخاص بالإجراء vsbRadius_Change().

وكما ترى، يعتبر التصريح عن متحول ساكن، حيلة جيدة للاستخدام، عندما لا ترغب بفقدان قيمة المتحول بعد الانتهاء من تنفيذ الإجراء.

تُخزن المتحولات غير الساكنة النظامية، والمتحولات الساكنة في الحاسب الشخصي PC بطرق مختلفة. فمثلاً قارن بين كيفية تخزين المتحول Radius، (والذي يمثل متحولاً نظامياً غير ساكن ومحلي ضمن الإجراء vsbRadius_Change()، وتخزين المتحول LastValue الساكن.

تضيع قيمة المتحول Radius بعد الخروج من الإجراء ()vsbRadius_Change إلى الأبد، وهكذا فإن خلايا الذاكرة التي كانت مخصصة لحفظ المتحول Radius تصبح حرة (أي خالية). أما خلايا الذاكرة المستخدمة لحفظ المتحول الساكن LastValue، تبقى محجوزة طالما أن برنامج الدائرة قيد التنفيذ. كذلك الأمر، يبقى المتحول المصرح عنه في قسم التصاريح العامة في الذاكرة طالما أن البرنامج يعمل.

رسم القطوع الناقصة والأقواس

تُستخدم الطريقة Circle لرسم القطوع الناقصة والأقواس.

وتعطى الصيغة العامة الكاملة لهذه الطريقة بما يلي:

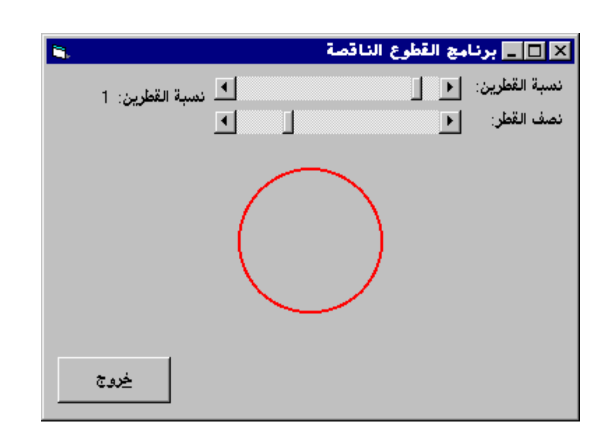
```
[object.]Circle [Step] ( X , Y ), _
                          radius , _
                          [color], _
                          [start], _
                          [ end ], _
                          [aspect]
```

يُؤدي استخدام الكلمة Step ضمن صيغة العبارة Circle، إلى الدلالة على أن الإحداثيين (X,Y) منسوبين للإحداثيين CurrentX , CurrentY. فمثلاً، إذا كانت قيمة CurrentX تساوي 1000، وقيمة CurrentY تساوي 3000، ترسم العبارة التالية دائرة بنصف قطر يساوي 80، ومركز الإحداثيات (X = 1010 , Y = 3010):

```
frmMyform.Circle Step (10,20),80
```

الوسيط Aspect (نسبة القطرين)

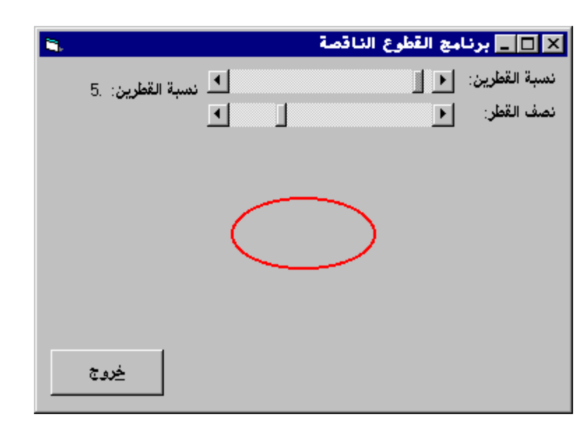
الوسيط Aspect عبارة عن رقم موجب يتسبب برسم قطوع ناقصة. فمثلاً، تُولد الطريقة Circle دائرة مكتملة إذا كانت قيمة Aspect تساوي الواحد، أما عندما تكون قيمة Aspect أكبر من الواحد، فتولد الطريقة Circle عند ذلك، قَطعاً ناقصاً، يتطاول على محوره العمودي. وعندما تكون قيمة Aspect أقل من الواحد، تولد الطريقة Circle قَطعاً ناقصاً يمتد على محوره الأفقي. تُبين الأشكال 8–14و 8–15و 8–16 ثلاثة قطوع ناقصة بثلاثة قيم مختلفة للوسيط Aspect (نسبة القطرين).



الشكل 8−14

رسم دائرة مكتملة

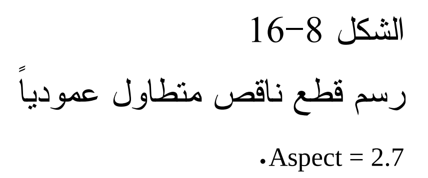
Aspect = 1•



الشكل 8−15

رسم قطع ناقص متطاول أفقياً

Aspect = 0.5•



الشكل 8−16

رسم قطع ناقص متطاول عمودياً

Aspect = 2.7•



برنامج القطوع الناقصة

يوضح برنامج القطوع الناقصة، كيفية رسم القطوع الناقصة، بأنصاف أقطار وقيم مختلفة للوسيط Aspect.

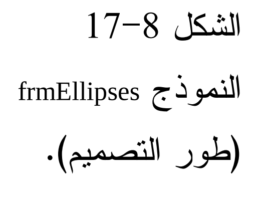
التمثيل المرئي لبرنامج القطوع الناقصة

نبدأ كعادتنا بالتمثيل المرئي لنموذج البرنامج:

- ❑أنشئ مشروعاً جديداً قياسياً من النوع Standard EXE، ثم احفظ نموذج المشروع بالاسم Ellipses.frm في الدليل C:\VB5Prg\Ch08، واحفظ ملف المشروع في ذات الدليل بالاسم Ellipses.vbp.
- ❑ أنشئ النموذج frmEllipses طبقاً للجدول 8-8.

يُفترض أن يبدو النموذج كما هو مبين في الشكل 8-17.




الشكل 8-17
النموذج frmEllipses
(طور التصميم).

الجدول 8-8. جدول خصائص النموذج frmEllipses.

الكائن	الخاصية	القيمة
Form	**Name**	**frmEllipses**
	Caption	برنامج القطوع الناقصة
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
Horizontal Scroll Bar	**Name**	**hsbAspect**
	Max	100
	Min	1
	Value	1
Horizontal Scroll Bar	**Name**	**hsbRadius**

الكائن	الخاصية	القيمة
	Max	100
	Min	1
	Value	1
Label	**Name**	**lblInfo**
	Caption	نسبة القطرين:
	RightToLeft	True
Label	**Name**	**lblAspect**
	Caption	نسبة القطرين:
	RightToLeft	True
Label	**Name**	**lblRadius**
	Caption	نصف القطر:
	RightToLeft	True

إدخال نص برنامج القطوع الناقصة

ستدخل الآن نص برنامج القطوع الناقصة:

- تحقق بأن العبارة Option Explicit تقع ضمن قسم التصاريح العامة للنموذج frmEllipses:

```
' يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء cmdExit_Click():

```
Private sub cmdExit_Click()
    End
End sub
```

- أدخل النص التالي ضمن الإجراء Form_Load():

```
Private Sub Form_Load()
    hsbAspect.Value = 10
    hsbRadius.Value = 10
    lblInfo.Caption = "نسبة القطرين: 1"
    frmEllipses.DrawWidth = 2
End Sub
```

❑ أدخل النص التالي ضمن الإجراء ()hsbAspect_Change:

```
Private Sub hsbAspect_Change()
    Dim X, Y
    Dim Info
    X = frmEllipses.ScaleWidth / 2
    Y = frmEllipses.ScaleHeight / 2
    frmEllipses.Cls
    frmEllipses.Circle (X, Y), hsbRadius.Value * 10, _
            RGB(255, 0, 0), , , hsbAspect.Value / 10
    Info = "نسبة القطرين: " + Str(hsbAspect.Value / 10)
    frmEllipses.lblInfo.Caption = Info
End Sub
```

❑ أدخل النص التالي ضمن الإجراء ()hsbAspect_Scroll:

```
Private Sub hsbAspect_Scroll()
    hsbAspect_Change
End Sub
```

❑أدخل النص التالي ضمن الإجراء ()hsbRadius_Change للنموذج frmEllipses:

```
Private Sub hsbRadius_Change()
    Dim X, Y
    Dim Info
    X = frmEllipses.ScaleWidth / 2
    Y = frmEllipses.ScaleHeight / 2
    frmEllipses.Cls
    frmEllipses.Circle (X, Y), hsbRadius.Value * 10, _
            RGB(255, 0, 0), , , hsbAspect.Value / 10
    Info = "نسبة القطرين: " + Str(hsbAspect.Value / 10)
    frmEllipses.lblInfo.Caption = Info
End Sub
```

❑ أدخل النص التالي ضمن الإجراء ()hsbRadius_Scroll:

```
private sub hsbRadius_Scroll()
    hsbRadius_Change()
End sub
```

تنفيذ برنامج القطوع الناقصة

لنشاهد ما كتبناه على أرض الواقع:

- نفّذ برنامج القطوع الناقصة.
- بدّل موقع مؤشر شريط التمرير **نصف القطر** لرسم دوائر وقطوع، بواسطة قيم مختلفة لنصف القطر.
- بدّل موقع مؤشر شريط التمرير **نسبة القطرين** لرسم قطوع ناقصة ذات قيم مختلفة للوسيط Aspect.

لاحظ أن قيمة الوسيط Aspect الراهنة تظهر على يمين شريط التمرير **نسبة القطرين**. يرسم البرنامج دائرة مكتملة عندما تكون قيمة **نسبة القطرين** تساوي الواحد، أما عندما تكون قيمة **نسبة القطرين** أقل من الواحد، يرسم البرنامج عندها قَطعاً ناقصاً ممتداً على محوره الأفقي، كما يظهر من الشكل 8–15، أما عندما تكون قيمة **نسبة القطرين** أكبر من الواحد، يرسم البرنامج قطعاً ممتداً على محوره العمودي، حسب ما يوضحه الشكل 8–16.

كيف يعمل برنامج القطوع الناقصة

يرسم البرنامج قطوعاً ناقصة باستخدام الطريقة Circle. وتستخدم الخاصية Value لشريطي التمرير كوسائط تمرر للطريقة Circle.

نص الإجراء Form_Load()

ينفذ الإجراء Form_Load() عند تحميل النموذج frmEllipses (أي في بداية التنفيذ):

```
Private Sub Form_Load()
   hsbAspect.Value = 10
   hsbRadius.Value = 10
   lblInfo.Caption = "نسبة القطرين: 1"
   frmEllipses.DrawWidth = 2
End Sub
```

يسند هذا الإجراء القيمة 10 إلى الخاصية Value لشريط التمرير ثم يظهر اللافتة lblInfo وهي تحمل العنوان (نسبة القطرين: 1)، على يمين شريط التمرير **نسبة القطرين**. ستلاحظ قريباً أن البرنامج يستخدم عُشْر قيمة الخاصية Value لشريط التمرير **نسبة القطرين**، كقيمة للوسيط Aspect في الطريقة Circle.

فمثلاً، عندما تكون قيمة الخاصية Value لشريط التمرير **نسبة القطرين** مساوية إلى 20، فهذا يعني أن الطريقة Circle تستخدم القيمة 2 كقيمة للوسيط Aspect، وهذا هو السبب وراء قيام الإجراء Form_Load() بتسمية اللافتة lblInfo بالعنوان (نسبة القطرين: 1) بعد إسناد القيمة 10 إلى شريط التمرير **نسبة القطرين**.

تسند آخر عبارة في هذا الإجراء القيمة 2 إلى الخاصية DrawWidth للنموذج.

```
frmEllipses.DrawWidth = 2
```

يتسبب هذا الأمر، برسم الأشكال بعرض يساوي إلى وحدتي قياس (أي 2 Units)، وحدة القياس قد تكون نقطة ضوئية Pixel، (حسبما هو محدد في الخاصية ScaleMode)، وهكذا فعند استخدام الطريقة Circle ترسم القطوع بخط عرضه وحدتين.

لاحظ أن الإجراء عمد إلى إسناد قيمة معينة للخاصية DrawWidth، وإسناد عنوان للافتة lblInfo، وإسناد قيمة أيضاً لشريطي التمرير، وهي عمليات كان يمكن إنجازها أثناء طور التصميم (أي في مرحلة التمثيل المرئي).

نص الإجراء hsbAspect_Change()

ينفذ الإجراء hsbAspect_Change() عند تبديل موقع شريط التمرير **نسبة القطرين**:

```
Private Sub hsbAspect_Change()
    Dim X, Y
    Dim Info
    X = frmEllipses.ScaleWidth / 2
    Y = frmEllipses.ScaleHeight / 2
    frmEllipses.Cls
    frmEllipses.Circle (X, Y), hsbRadius.Value * 10, _
            RGB(255, 0, 0), , , hsbAspect.Value / 10
    Info = "نسبة القطرين: " + Str(hsbAspect.Value / 10)
    frmEllipses.lblInfo.Caption = Info
```

```
End Sub
```

يحسب هذا الإجراء إحداثي مركز النموذج ويسندهما للمتحولين X,Y.

ينظف الإجراء سطح النموذج من أي قطوع مرسومة مسبقاً، مستخدماً الطريقة Cls، ثم يرسم القطع الناقص:

```
frmEllipses.Circle (X, Y), hsbRadius.Value * 10, _
          RGB(255, 0, 0), , , hsbAspect.Value / 10
```

يُسند إلى الوسيط Radius (نصف القطر) القيمة الحالية لشريط التمرير **نصف القطر** مضروبة بـــ 10، كما يُسند إلى الوسيط Aspect، عُشْر قيمة الخاصية Value لشريط التمرير **نسبة القطرين**.

لاحظ أننا لم نستخدم الوسيطين الاختياريين Start وEnd للطريقة Circle بعد، بل وضعنا مكانهما فاصلتين (أي فاصلتين بين الوسيط Color والوسيط Aspect)، تشير هاتان الفاصلتان لفيجول بيسك بأن الوسيطين الاختياريين غير مذكورين (مهملين).

نص الإجراء hsbRadius_Change()

يُنفذ هذا الإجراء آلياً، عند تبديل موقع مؤشر شريط التمرير **نصف القطر** (أي تحريك الزالقة)، يعتبر هذا الإجراء مشابهاً للإجراء hsbAspect_Change()، فيظهر القطع الناقص وتعدّل محتويات اللافتة lblInfo.

لاحظ أن نص الإجراء hsbRadius_Scroll()، يستدعي الإجراء hsbRadius_Change()، كما أن نص الإجراء hsbAspect_Scroll() يستدعي بدوره الإجراء hsbAspect_Change()، (بمعنى أنك تنفذ الإجراءات المرافقة للحادثة تغيير Change عند حصول حادثة التمرير Scroll).

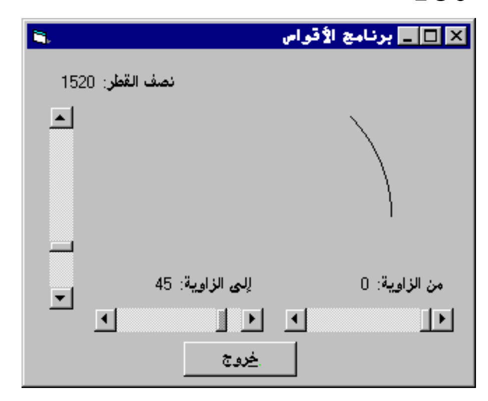
برنامج الأقواس

سنكتب الآن برنامج الأقواس Arcs، يوضح هذا البرنامج كيفية استخدام الطريقة Circle لرسم أقواس ذات نقاط بدء وانتهاء مختلفة. نذكرك الآن بالصيغة العامة الكاملة للطريقة Circle:

```
[object.]Circle [Step] ( X , Y ), _
```

```
radius , _
[color], _
[start], _
[ end ], _
[aspect]
```

يُحدد الوسيطان End, Start نقطتي البداية والنهاية لقوس الدائرة. فإذا كانت نقطة البداية تُساوي مثلاً الدرجة صفرْ، ونقطة النهاية تساوي الدرجة 45°، فهذا يعني أن هذا الجزء فقط من الدائرة هو الذي سيرسم. يوضح الشكل 8–18 جزء من دائرة (قوس) مرسوم بالطريقة Circle، نقطة بدايته الزاوية صفرْ ونقطة نهايته الزاوية 45°. كما يوضح الشكل 8–19 قوساً ذو نقطة بداية عند الدرجة 23° ونقطة نهاية عند الدرجة 180°.



الشكل 8–18
رسم قوس (0°–45°).



الشكل 8–19
رسم قوس (23°–180°).

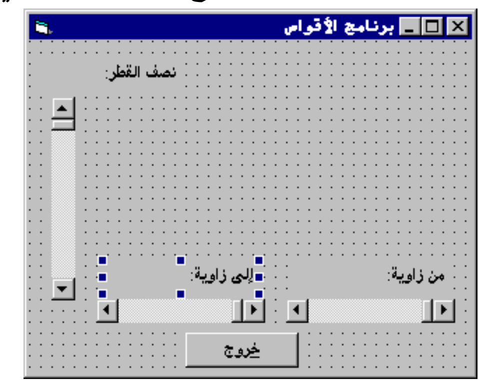
التمثيل المرئي لبرنامج الأقواس

سنصمم الآن نموذج برنامج الأقواس:

□أنشئ مشروعاً تنفيذياً قياسياً Standard EXE، ثم احفظ نموذج المشروع بالاسم Arcs.frm في الدليل C:\VB5Prg\Ch08، واحفظ ملف المشروع في ذات الدليل بالاسم Arc.Vbp.

□أنشئ النموذج frmArcs حسب الجدول 8-9.

يفترض أن يبدو النموذج المكتمل كما في الشكل 8-20.



الشكل 8-20

النموذج frmArcs.

في مرحلة التصميم.

الجدول 8-9. جدول خصائص النموذج frmArcs.

الكائن	الخاصية	القيمة
Form	**Name**	**frmArcs**
	Caption	برنامج الأقواس
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
Horizontal Scroll Bar	**Name**	**hsbFrom**
	Max	360
	Min	0
	Value	0
Horizontal Scroll Bar	**Name**	**hsbTo**
	Max	360
	Min	0

	Value	0
Vertical Scroll Bar	**Name**	**vsbRadius**
	Max	100
	Min	1
	Value	1
Label	**Name**	**lblFrom**

الكائن	الخاصية	القيمة
	Caption	من الزاوية:
	RightToLeft	True
Label	**Name**	**lblTo**
	Caption	إلى الزاوية:
	RightToLeft	True
Label	**Name**	**lblRadius**
	Caption	نصف القطر:
	RightToLeft	True

إدخال نص برنامج الأقواس

سنبدأ الآن بكتابة نص برنامج الأقواس:

- تحقق بأن قسم التصاريح العامة للنموذج frmArcs يحوي العبارة Option Explicit.

```
' يجب التصريح عن كل المتحولات
Option Explicit
```

- أنشئ إجراءاً جديداً يدعى DrawArcs في قسم التصاريح العامة للنموذج frmArcs. (انقر نقرةً مزدوجة على النموذج لإظهار نافذة نص البرنامج ثم اختر **Add Procedure** من القائمة **Tools**، ثم اختر النوع Sub في الحقل Type لمربع الحوار Add Procedure، وعين المجال Public في الحقل Scope، ثم اكتب DrawArc في المربع Name، وأخيراً انقر الزر **OK**).

❑ أدخل النص التالي ضمن الإجراء DrawArc() الذي أنشأناه في الخطوة السابقة:

```
Public Sub DrawArc()
    Dim x, y
    Const pi = 3.14159265
    x = frmArcs.ScaleWidth / 2
    y = frmArcs.ScaleHeight / 2
    frmArcs.Cls
    Circle (x, y), vsbRadius.Value * 20, , _
               hsbFrom * 2 * pi / 360, hsbTo * 2 * pi / 360
    lblFrom.Caption = "من الزاوية: " + Str(hsbFrom.Value)
    lblTo.Caption = "إلى الزاوية: " + Str(hsbTo.Value)
    lblRadius.Caption = "نصف القطر:" + Str(vsbRadius.Value * 20)
End Sub
```

❑ أدخل النص التالي ضمن الإجراء cmdExit_Click() للنموذج frmArcs:

```
Privat sub cmdExit_Click()
    End
End sub
```

❑ أدخل النص التالي ضمن الإجراء hsbFrom_Change() للنموذج frmArcs:

```
Privat sub hsbFrom_Change()
    DrawArc
End sub
```

❑ أدخل النص التالي ضمن الإجراء hsbFrom_Scroll() للنموذج frmArcs:

```
Privat sub hsbFrom_Scroll()
    hsbFrom_Change
End sub
```

❑ أدخل النص التالي ضمن الإجراء hsbTo_Change():

```
Privat sub hsbTo_Change()
    DrawArc
End sub
```

❑ أدخل النص التالي ضمن الإجراء hsbTo_Scroll للنموذج frmArcs:

```
Private Sub hsbTo_Scroll()
    hsbTo_Change
End Sub
```

❑ أدخل النص التالي ضمن الإجراء VsbRadius_Change للنموذج frmArcs:

```
Private Sub vsbRadius_Change()
```

```
    DrawArc
End Sub
```

- أدخل النص التالي ضمن الإجراء VsbRadius_Scroll للنموذج frmArcs:

```
Private Sub vsbRadius_Scroll()
    vsbRadius_Change
End Sub
```

- احفظ المشروع بالطريقة المعتادة.

تنفيذ برنامج الأقواس

لنشاهد برنامج الأقواس على أرض الواقع:

- نفّذ برنامج الأقواس.
- زد نصف القطر، بتحريك مؤشر شريط التمرير **نصف القطر** (شريط التمرير العمودي).
- زد قيمة شريط التمرير **من زاوية** (شريط التمرير الأفقي اليميني).
- زد قيمة شريط التمرير **إلى زاوية** (شريط التمرير الأفقي اليساري).

كما تلاحظ، يُرسم قوس يبدأ من النقطة المحددة بواسطة شريط التمرير **من زاوية**، وينتهي بالنقطة المحددة بشريط التمرير **إلى زاوية**.

يوضح الشكل 8-21 قوساً يبدأ من الدرجة 180°، وينتهي بالدرجة 23°.



الشكل 8-21

رسم قوس (180°-23°).

- انقر الزر **خروج** لإنهاء البرنامج.

كيف يعمل برنامج الأقواس

يَستخدم برنامج الأقواس الطريقة Circle لرسم الأقواس.

تُستخدم قيم الخاصية Value لكل من شريط التمرير **نصف القطر** وشريطي التمرير **من زاوية** و **إلى زاوية** كوسائط للطريقة Circle.

نص الإجراء DrawArc()

ينفذ الإجراء DrawArc() عند تغيير موقع مؤشر شريط التمرير **نصف القطر**، أو موقع مؤشر شريط التمرير **من زاوية**، أو عند تغيير موقع مؤشر شريط التمرير **إلى زاوية**.

يُعرّف هذا الإجراء قيمة الثابت pi (π)، ويُسند له القيمة العددية المكافئة:

```
CONST pi = 3.14159265
```

ثم يحسب إحداثيات مركز النموذج:

```
x = frmArcs.ScaleWidth / 2
y = frmArcs.ScaleHeight / 2
```

وينظف محتويات النموذج بعد ذلك بالطريقة Cls:

```
frmArcs.Cls
```

ثم يرسم القوس باستخدام الطريقة Circle:

```
Circle (x, y), vsbRadius.Value * 20, , _
           hsbFrom * 2 * pi / 360, hsbTo * 2 * pi / 360
```

نكرر الصيغة الكاملة للطريقة Circle:

```
[Object].Circle [Step] (X,Y),Radius, _
                        [color],[Start],[End], [Aspect]
```

يتحدد مركز القوس في هذا الإجراء بالوسيطين X,Y، كما يتحدد نصف قطر القوس بقيمة الخاصية Value لشريط التمرير vsbRadius مضروبة بـــ 20.

تُحدد قيمة الخاصية Min لشريط التمرير، أثناء مرحلة التصميم بالقيمة 1، كما حُددت الخاصية Max لهذا الشريط بالقيمة 100. مما يعني أن نقل مؤشر شريط التمرير من الموضع الأدنى Min إلى الموضع الأعلى Max، يستلزم النقر مائة مرة على أسهم الشريط.

تُضرب قيمة Value بالقيمة 20، ثم تُستخدم القيمة الناتجة للوسيط Radius (نصف القطر)، مما يعني أن نصف القطر يمكن أن يتخذ أي قيمة ما بين 20 و2000.

لم نستخدم في مثالنا هذا الوسيط Color، وإنما وضعنا فاصلة مكانه للإشارة إلى عدم استخدامه. يستخدم البرنامج في مثل هذه الحالة اللون المحدد بالخاصية ForeColor للنموذج، والذي يكون اللون الأسود في الحالة الافتراضية.

الوسيطان التاليان في الطريقة Circle هما: Start و End، ويجب تمرير قيمتهما بالراديان.

تم إسناد القيمة 1 للخاصية Min لشريطي التمرير **من زاوية** و **إلى زاوية**، وإسناد القيمة 360 للخاصية Max. مما يعني أن كلا الشريطين، مقسم إلى 360 جزء، يمثل كل جزء منها درجة واحدة، وتستخدم الصيغة التالية للتحويل من الدرجات إلى الراديان:

```
Radians = Degree * 2 * pi / 360
```

فمثلاً، الدرجة 360 تكافئ:

```
360 * 2 * pi / 360 = 6.2831853 Radians
```

وهذا ما يشرح السبب الذي دفع الإجراء DrawArc إلى جداء قيمة الخاصية Value للشريطين **من زاوية** (يمثل الوسيط Start)، والشريط **إلى زاوية** (يمثل الوسيط End) بالقيمة $\frac{2*\Pi}{360}$.

لم يُستخدم الوسيط Aspect الاختياري هنا، وبما أن هذا الوسيط آخر وسيط Circle فلا توجد حاجة لوضع فاصلة بعد الوسيط End.

ملاحظة

يمكن الحصول على القيمة الحقيقة للثابت π، عن طريق العبارة التالية:

```
Dim pi As Double
pi = 4 * ATN(1)
```

آخر شيء ينجزه هذا الإجراء، هو إظهار المحتويات الجديدة للافتات: **نصف القطر** و **من زاوية** و **إلى زاوية**:

```
lblFrom.Caption = "من الزاوية:" + Str(hsbFrom.Value)
```

```
lblTo.Caption = ":إلى الزاوية " + Str(hsbTo.Value)
lblRadius.Caption = ":نصف القطر" + Str(vsbRadius.Value * 20)
```

تمكّنك هذه العبارات الثلاث من مشاهدة القيم الراهنة لأشرطة التمرير.

نص الإجراء hsbFrom_Change()

يُنفَذ هذا الإجراء آلياً، عند تغيير موقع مؤشر شريط التمرير **من زاوية**:

```
Private Sub hsbFrom_Change()
    DrawArc
End Sub
```

يُنفذ هذا الإجراء DrawArc() الذي يرسم القوس، تبعاً للقيمة الجديدة لشريط التمرير **من زاوية**.

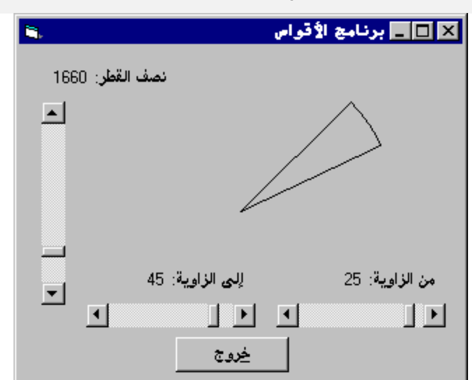
يُنفِذ كلا الإجرائين hsbTo_Change() و vsbRadius_Change() الإجراء DrawArc()، عند تغيير موقع مؤشر أحد الشريطين: **إلى زاوية** أو **نصف القطر**.

المزيد عن الوسيطين Start و End للطريقة Circle

يسمح هذان الوسيطان باستخدام قيم سالبة. وعند تمرير قيم سالبة للوسيط Start، يرسم فيجول بيسك خطاً مستقيماً من مركز القوس إلى نقطة بداية القوس، كما أن تمرير قيم سالبة للوسيط End، تتسبب برسم خط مستقيم من مركز القوس إلى نقطة نهاية القوس.

يوضح الشكل 8-22 قوساً يرسم بالعبارة التالية:

```
Circle (X,Y),1000, -25 * 2 * pi / 360, -45 * 2 * pi / 360
```

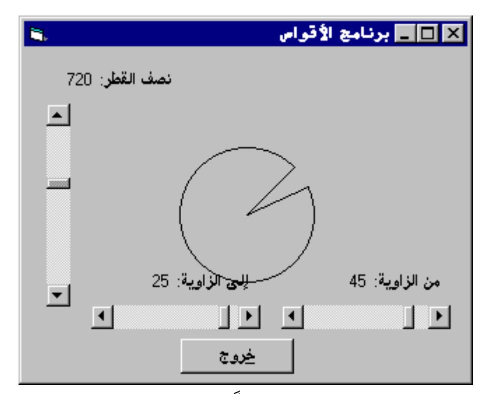


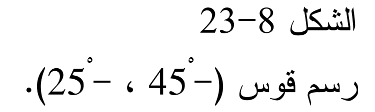
الشكل 8-22

رسم قوس (-25°، -45°).

كما يبين الشكل 8-23 قوساً رسم بواسطة العبارة التالية:

```
Circle (X,Y),1000, -45 * 2 * pi / 360, -25 * 2 * pi / 360
```

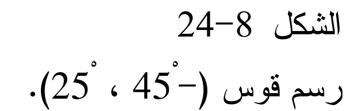



الشكل 8-23
رسم قوس (-°45 ، -°25).

ويوضح الشكل قوساً رسم بالعبارة التالية:

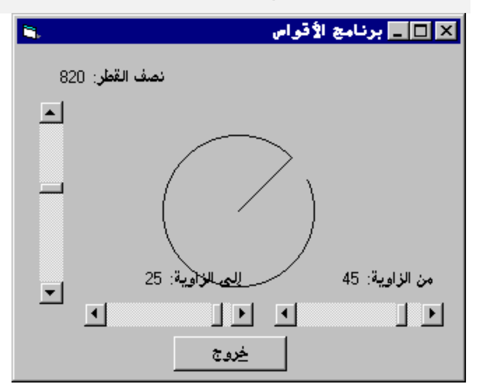
```
Circle (X,Y),1000, -45 * 2 * pi / 360, 25 * 2 * pi / 360
```

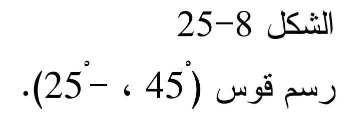



الشكل 8-24
رسم قوس (-°45 ، °25).

ويبين الشكل 8-25 قوساً رُسم بواسطة العبارة التالية:

```
Circle (X,Y),1000, 45 * 2 * pi / 360, -25 * 2 * pi / 360
```




الشكل 8-25
رسم قوس (°45 ، -°25).

يُمكنك التمرين على استخدام الوسيطين Start و End للطريقة Circle، المستخدمة ضمن الإجراء ()DrawArc كما يلي:

- غيّر الطريقة Circle في الإجراء ()DrawArc، إلى العبارة التالية، كي تتمكن من

رسم أقواس مشابهة لتلك المبين في الشكلين 8−22 و8−23:

```
Circle (X, Y),vsbRadius.Value * 20, _
             -hsbFrom * 2* pi / 360, -hsbTo * 2 * pi / 360
```

❑غيّر الطريقة Circle إلى العبارة التالية، حتى تتمكن من رسم أقواس مشابهة للمبينة في الشكل 8−24:

```
Circle (X, Y),vsbRadius.Value * 20, _
             -hsbFrom * 2* pi / 360, hsbTo * 2 * pi / 360
```

❑ أما لرسم أقواس تشبه القوس المبينة في الشكل 8−25 فاستخدم العبارة التالية:

```
Circle (X, Y),vsbRadius.Value * 20, _
             hsbFrom * 2* pi / 360, -hsbTo * 2 * pi / 360
```

الخاصية AutoRedraw

تسمح هذه الخاصية بإعادة رسم محتويات الكائن الرسومية، في حال تم تغطية هذا الكائن بكائن آخر مثلاً. والقيمة الافتراضية لها تساوي False.

استخدم الخطوات التالية لرؤية تأثير الخاصية AutoRedraw:

❑ نفّذ برنامج الأقواس.

❑ارسم قوساً بتغيير مواقع مؤشرات أشرطة التمرير: **من زاوية** و **إلى زاوية** و **نصف القطر**.

❑صغّر إطار برنامج الأقواس بنقر رمز إشارة الناقص، الذي يظهر عند الزاوية اليمنى العليا من إطار برنامج الأقواس.

يستجيب البرنامج بتصغير إطار برنامج الأقواس ويظهره كرمز في شريط المهام.

❑ استرجع إطار برنامج الأقواس إلى حجمه الطبيعي.

ستجد أن القوس المرسومة لم تظهر في الإطار، والسبب أن قيمة الخاصية AutoRedraw تساوي False.

❑أنهِ البرنامج، ثم أسند القيمة True إلى الخاصية AutoRedraw، وكرر الخطوات السابقة.

ستلاحظ هذه المرة أن البرنامج أعاد رسم القوس آلياً، والسبب في ذلك، أن قيمة الخاصية AutoRedraw تساوي True هذه المرة.

الإجراء ()Form_Paint

تستطيع إظهار هذا الإجراء، بالنقر المزدوج على النموذج، والانتقال إلى إطار نص البرنامج، ثم اختيار Form من مربع السرد اليساري، واختيار Paint من مربع السرد اليميني.

استخدم هذا الإجراء لرسم خمسة خطوط أفقية كما يلي:

- أرجع الخاصية AutoRedraw إلى القيمة False مرة أخرى.
- أضف السطور التالية إلى الإجراء ()Form_Paint:

```
Private Sub Form_Paint()
    frmArcs.Line (2000, 100)-(4000, 100)
    frmArcs.Line (2000, 200)-(4000, 200)
    frmArcs.Line (2000, 300)-(4000, 300)
    frmArcs.Line (2000, 400)-(4000, 400)
    frmArcs.Line (2000, 500)-(4000, 500)
End Sub
```

- احفظ المشروع باختيار **Save Project** من القائمة **File** في فيجول بيسك.
- نفّذ برنامج الأقواس، وشاهد الخطوط الأفقية الخمسة، وقد رسمت على سطح برنامج الأقواس.
- غطِ إطار برنامج الأقواس، بإطار تطبيق آخر، (المهم تغطية الخطوط الأفقية الخمس)، ثم ارفع إطار ذلك التطبيق عنه مرة ثانية.

كما تلاحظ، أُعيد رسم الخطوط مرة ثانية.

يُنفذ الإجراء ()Form_Paint، كلما احتاج فيجول بيسك إلى إعادة رسم النموذج أو جزء منه، وبما أن فيجول بيسك يرسم النموذج عند بداية تشغيل البرنامج، فهذا يعني أن الإجراء ()Form_Paint ينفذ آلياً عند بدء تشغيل البرنامج. كما يُنفذ فيجول بيسك هذا الإجراء كلما احتاج إلى إعادة رسم النموذج أو جزء منه كما ذكرنا سابقاً.

وبشكل عام، يمكنك الاستفادة من الإجراء Form_Paint()، مثل الاستفادة الحاصلة من إسناد True للخاصية AutoRedraw.

تمتاز الخاصية AutoRedraw بسهولة العمل، فلا حاجة عندها لكتابة نصوص برنامج لإعادة الرسم مرة أخرى. إلا أن سلبية الخاصية AutoRedraw في أنها تستهلك قدراً لا بأس به من الذاكرة، وتتسبب بفترات تأخير ملحوظة أثناء تنفيذ البرنامج، وخصوصاً على الأجهزة البطيئة نسبياً.

الخلاصة

تعلمت فى هذا الفصل كيفية استخدام أساليب الرسم المختلفة، والكثير عن وسائطها، وذلك لرسم أشكال معقدة نوعاً ما، مثل الدوائر والأقواس والقطاعات الناقصة، بالإضافة طبعاً إلى رسم النقاط والخطوط.

الفصل التاسع

إظهار البيانات بصيغة مجدولة

تدعو الضرورة أحياناً إلى إظهار البيانات بشكل صفوف وأعمدة (أي ضمن جداول). يمكن إنجاز ذلك باستخدام الطريقة Print عن طريق إظهار النص سطراً تلو الآخر وحساب المواقع التي ستظهر البيانات عندها.

وعموماً، يقدم فيجول بيسك عنصر تحكم الجدول، يمكّنك من تشكيل الجداول بسهولة. سنتعلم في هذا الفصل كيفية استخدام عنصر تحكم الجدول.

برنامج الجدول

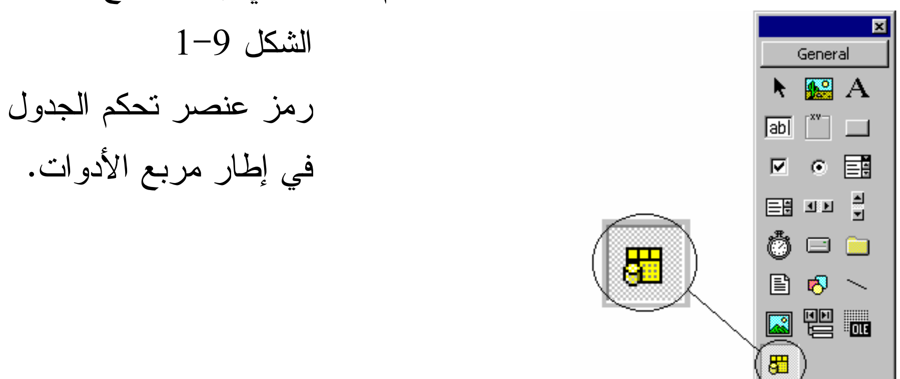
سنكتب الآن برنامجاً ندعوه برنامج الجدول، الذي يعتبر مثالاً على كيفية استخدام عنصر تحكم الجدول، اللازم لإنشاء جدول من المعلومات.

- أنشئ الدليل C:\VB5Prg\Ch09، لأننا سنستخدمه لحفظ برامج هذا الفصل.

❑أنشئ مشروعاً تنفيذياً قياسياً Standard EXE، واحفظ نموذج المشروع بالاسم Table.frm في الدليل C:\VB5Prg\Ch09، واحفظ ملف المشروع بالاسم Table.Vbp في ذات الدليل.

سنتأكد أولاً من وجود عنصر تحكم الجدول في المشروع قبل استخدامه، علماً بأن عنصر تحكم الجدول، عبارة عن عنصر من النوع OCX.

يوضح الشكل 9-1 رمز عنصر تحكم الجدول. ضع مؤشر الفأرة (بدون النقر عليها) فوق رمز عنصر التحكم هذا، فيظهر مستطيل أصفر يحمل الرسالة MsFlexGrid، وبهذه الطريقة تتحقق من وجود رمز عنصر تحكم الجدول في إطار مربع الأدوات.



الشكل 9-1

رمز عنصر تحكم الجدول

في إطار مربع الأدوات.

إذا لم يكن عنصر تحكم الجدول موجوداً في إطار مربع الأدوات، أضفه باتباع ما يلي:

❑ اختر **Components** من القائمة **Project** لفيجول بيسك، واختر الصفحة Controls لمشاهدة عناصر التحكم المتاحة (Controls).

يستجيب فيجول بيسك بإظهار صفحة عناصر التحكم Controls في مربع الحوار Components.

❑ تأكد أن خانة الاختيار التي تظهر في الصفحة Control بجانب البند Selected Item Only لا تحوى علامة اختيار داخلها، (لأنك ترغب بمشاهدة لائحة عناصر التحكم الكلية، وليس فقط العناصر المحتواة مسبقاً في المشروع).

▪تفحّص لائحة عناصر التحكم المخصصة حتى تجد البند Microsoft

FlexGrid Control 5.0.

▪عند وجود علامة اختيار يسار البند Microsoft FlexGrid Control 5.0، فهذا يعني أن عنصر تحكم الجدول موجود مسبقاً في إطار مربع الأدوات، ثم انقر على الزر **OK** في مثل هذه الحالة، للخروج من مربع الحوار Components.

▪إذا لم يكن هناك علامة اختيار يسار هذا البند، فهذا يعني أن عنصر التحكم ليس موجوداً حالياً في إطار مربع الأدوات، ولإضافته انقر على خانة الاختيار بجانبه لاختياره، ثم انقر الزر **OK**.

اتبع الخطوات التالية لإضافة البند Microsoft FlexGrid Control 5.0، إذا لم يظهر في اللائحة السابقة:

- انقر الزر **Browse** في الصفحة Controls في مربع الحوار Components.

يظهر نتيجة ذلك مربع الحوار Add ActiveX Control.

- اختر الملف MsFlxGrd.OCX من الدليل System ثم انقر على الزر **فتح**.

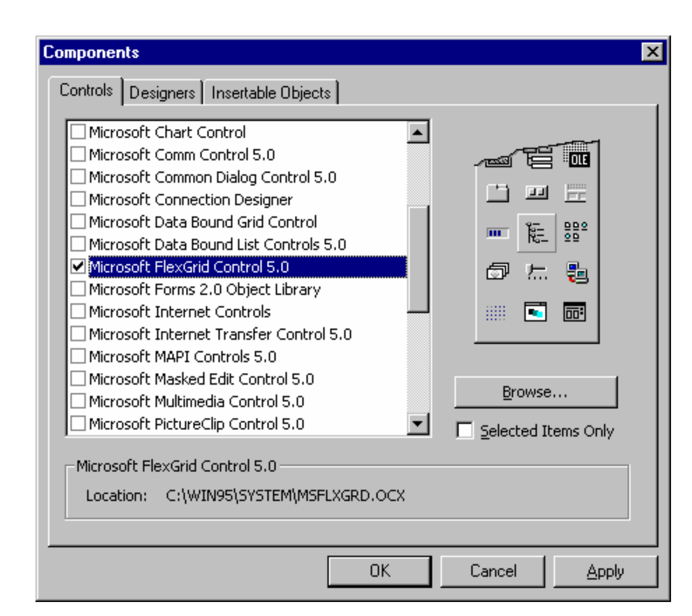
عندها يظهر عنصر التحكم Microsoft FlexGrid Control 5.0 نتيجة ذلك، كأحد بنود لائحة عناصر التحكم Controls.

- تحقق من وجود علامة اختيار في خانة الاختيار يسار عنصر التحكم Microsoft FlexGrid Control 5.0 ثم انقر الزر **OK**.

يُظهر الشكل 9-2 البند Microsoft FlexGrid Control 5.0 ضمن الصفحة Controls للإطار Components.

ملاحظة

إذا كنت تعمل ضمن النظام Windows95، يكون اسم دليل النظام هو:
C:\Windows\System
أما إذا كنت تعمل ضمن النظام WinNT، يكون اسم دليل النظام هو:
C:\WinNT\System32



الشكل 9-2

البند Microsoft FlexGrid Control 5.0.

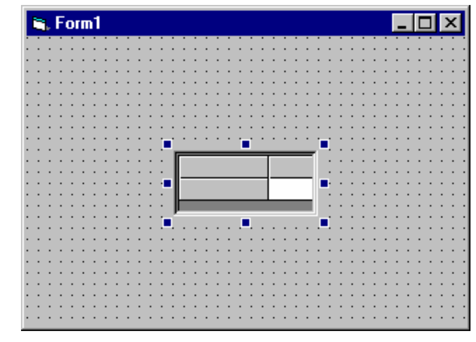
ملاحظة

يأتي فيجول بيسك بعدة إصدارات منها مثلاً Learning Edition و Professional Edition و Enterprise Edition.

فإذا لم تحتوِ إصداراتك على الملف MsFlxGrd.OCX، فهذا يعني أنك لن تقدر على بناء برنامج الجدول (لأن برنامج الجدول سوف يستخدم عنصر التحكم OCX هذا). لكن رغم ذلك استمر في قراءة برنامج الجدول حتى يتضح لك كيفية استخدام عنصر تحكم الجدول ضمن البرامج.

- والآن انقر نقرة مزدوجة فوق رمز عنصر الجدول في مربع الأدوات.

يظهر عنصر الجدول نتيجة ذلك ضمن النموذج، حسب ما يبينه الشكل 9-3.



الشكل 9-3

وضع عنصر الجدول

في النموذج (طور التصميم).

أنشئ النموذج بإسناد الخصائص التالية له:

```
Name:            frmTable
```

```
Caption:        برنامج الجدول
RightToLeft:    True
```

أسند لعنصر تحكم الجدول الخصائص التالية:

```
Name:           grdTable
Rows:           13
Cols:           5
RightToLeft:    True
```

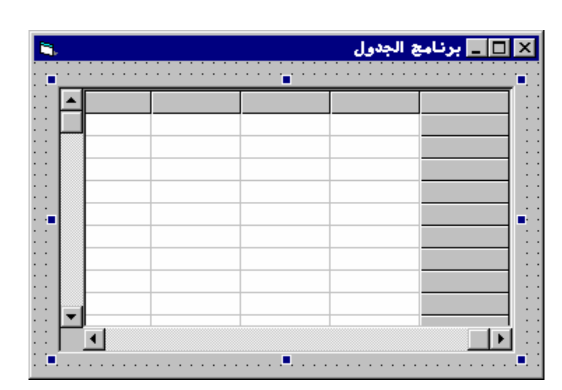
يفترض أن يبدو الجدول المكبر كما في الشكل 9-4.

- كبّر النموذج frmTable، وضع فيه زراً آخر.
- أسند لزر الأمر الجديد الخصائص التالية:

```
Name:           cmdExit
Caption:        &خروج
RightToLeft:    True
```

ملاحظة

تم إسناد القيمة True للخاصية RightToLeft التابعة لعنصر تحكم الجدول، وذلك بغية تعريبه، وبذلك يصبح العمود الأول من جهة اليمين بدلاً من جهة اليسار، بالنسبة للمستخدم. اقرأ الفصل 22 لمزيد من الإيضاح.



الشكل 9-4

عنصر الجدول بعد تكبيره.

- تحقق من وجود العبارة Option Explicit ضمن قسم التصاريح العامة:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء Form_Load() للنموذج frmTable:

```
Private Sub Form_Load()
```

```
'اجعل السطر الحالي هو السطر رقم صفر'
grdTable.Row = 0

'اكتب في الخلية (سطر=0،عمود=1)'
grdTable.Col = 1
grdTable.Text = "كهرباء"

grdTable.Col = 2
grdTable.Text = "ماء"

grdTable.Col = 3
grdTable.Text = "مواصلات"

grdTable.Col = 4
grdTable.Text = "طعام"

grdTable.Col = 0

grdTable.Row = 1
grdTable.Text = "كانون الثاني"

grdTable.Row = 2
grdTable.Text = "شباط"

grdTable.Row = 3
grdTable.Text = "آذار"

grdTable.Row = 4
grdTable.Text = "نيسان"

grdTable.Row = 5
grdTable.Text = "أيار"

grdTable.Row = 6
grdTable.Text = "حزيران"

grdTable.Row = 7
grdTable.Text = "تموز"
```

```
    grdTable.Row = 8
    grdTable.Text = "آب"

    grdTable.Row = 9
    grdTable.Text = "أيـلـول"

    grdTable.Row = 10
    grdTable.Text = "تـشـريـن الأول"

    grdTable.Row = 11
    grdTable.Text = "تـشـريـن الـثـانـي"

    grdTable.Row = 12
    grdTable.Text = "كـانـون الأول"

End Sub
```

❑ أدخل النص التالي ضمن الإجراء cmdExit_Click() للنموذج frmTable:

```
Private Sub cmdExit_Click()
    End
End Sub
```

❑ احفظ المشروع بالطريقة المعتادة.

❑نفّذ البرنامج الآن، رغم أننا لم ننته بعد من كتابته، يُظهر البرنامج، عنصر الجدول كما في الشكل 9-5.



الشكل 9-5

برنامج الجدول.

❑ تنقل بواسطة مفاتيح الأسهم من خلية إلى أخرى عبر الجدول.

كما تلاحظ، يمتلك عنصر الجدول الآن 13 سطراً و5 أعمدة (بما في ذلك سطر العنوان وعمود العنوان). والسبب في ذلك طبعاً، هو إسناد القيمة 13 إلى الخاصية Rows، وإسناد القيمة 5 إلى الخاصية Cols للعنصر grdTable.

- أنهِ البرنامج بنقر الزر **خروج**.

نفّذ الخطوتين التاليتين وأعد تشغيل البرنامج، وراقب النتائج:

- أسند القيمة 2-flexGridInset للخاصية GridLines للعنصر grdTable.
- أسند اللون الأبيض White إلى الخاصية BackColor (الخلفية) للعنصر grdTable.

نص الإجراء Form_Load()

يُنفّذ هذا الإجراء عند بدء تحميل النموذج frmTable. يبدأ هذا الإجراء بتحديد السطر الراهن (عن طريق إسناد القيمة صفر للخاصية Row):

```
' اجعل السطر الحالي هو السطر رقم صفر
grdTable.Row = 0
```

ملاحظة

يمتلك عنصر الجدول، خاصية تدعى Row، وأخرى تدعى Rows. حاول جاهداً أن لا تخلط بينهما.

تستخدم الخاصية Row لمعرفة أو تغيير السطر الحالي، بينما تستخدم الخاصية Rows لمعرفة أو تغيير عدد الأسطر الحالي.

كما يمتلك عنصر الجدول خاصية تدعى Col، وأخرى تدعى Cols، تجنب كذلك الخلط بينهما.

تستخدم الخاصية Col لمعرفة أو تغيير العمود الحالي، بينما تستخدم الخاصية Cols

لمعرفة أو تغيير عدد الأعمدة الحالي.

يكتب الإجراء حال تحديد السطر الراهن، ضمن الخلية المحددة بالسطر Row # 0 والعمود Col # 1 كما يلي:

```
'اكتب في الخلية (سطر=0، عمود=1)
grdTable.Col = 1
grdTable.Text = "كهرباء"
```

أي بمعنى أن الإجراء حدد السطر الراهن (السطر الفعال) على أنه السطر رقم صفر، وحدد العمود رقم 1 على أنه العمود الفعال، ثم أسند العنوان **كهرباء**، إلى الخاصية Text لعنصر الجدول، ونتيجة ذلك، تتوضع كلمة **كهرباء** في السطر رقم صفر، والعمود رقم 1.

بشكل مشابه، يُسند الإجراء العناوين: **ماء** و **مواصلات** و **طعام**، إلى الخلايا المحددة بالسطر صفر والعمود 2 والعمود 3 والعمود 4 على التوالي. فمثلاً، استخدمت العبارات التالية لتعيين الخاصية Text المحددة بالسطر صفر والعمود 4.

```
grdTable.Col = 4
grdTable.Text = "طعام"
```

لا حاجة طبعاً لإسناد القيمة صفر مجدداً إلى الخاصية Row، كلما غيرنا قيمة الخاصية Col، لأنها تحافظ على قيمتها، ما لم يتم إسناد قيمة أخرى إليها.

ملاحظة

يبدأ ترقيم الأسطر والأعمدة في عنصر الجدول، بدءاً من الرقم صفر، وبذلك يكون رقم السطر للخلية الأولى صفراً، ورقم عمودها صفر أيضاً.

يُباشر الإجراء حال الانتهاء من كتابة العناوين الأربعة الرأسية، بتحديد الخاصية Text للعمود اليساري (العناوين الجانبية). فمثلاً استخدمت العبارات التالية لتحديد الخاصية Text للخلية المحددة بالسطر 1 والعمود صفر:

```
grdTable.Col = 0
grdTable.Row = 1
grdTable.Text = "كانون الثاني"
```

ملاحظة

يُظهر عنصر الجدول، المعلومات بصيغة جدولية، حيث تستطيع الانتقال من خلية لأخرى، إما بواسطة مفاتيح الأسهم، أو بواسطة شريطي التمرير، لكن لا يمكنك إدخال المعلومات بواسطة لوحة المفاتيح مباشرة داخل الخلية.

تغيير عرض الخلية

قد لا تكون الخلية بالاتساع الكافي، وذلك تبعاً للخط المستخدم، فمثلاً قد لا تتسع الكلمة **كهرباء**، ضمن خليتها (انظر إلى الشكل 9–5). تستطيع اتباع الخطوات التالية لتوسيع الخلية أثناء زمن التنفيذ.

- أضف إجراءً جديداً للنموذج frmTable بالنقر المزدوج عليه (لإظهار إطار نص البرنامج)، ثم باختيار **Add Procedure** من القائمة **Tools** لفيجول بيسك.

يُظهر فيجول بيسك مربع الحوار Add Procedure.

  - أدخل اسم الإجراء الجديد في الحقل Name، لنفترض أننا سنسمي هذا الإجراء SetColWidth.
  - تحقق من اختيار Sub في مربع الحوار Add Procedure.
  - تحقق من اختيار Public في مربع الحوار Add Procedure.
  - انقر الزر **OK** في مربع الحوار Add Procedure.

يُشكل فيجول بيسك إجراءً جديداً يدعى SetColWidth، ويُظهره في المنطقة العامة General للنموذج frmTable.

- أدخل النص التالي ضمن الإجراء SetColWidth:

```
Public Sub SetColWidth()
    Dim Counter
    For Counter = 0 To 4 Step 1
      grdTable.ColWidth(Counter) = 1300
    Next
End Sub
```

يستخدم هذا الإجراء الحلقة For لتبديل الخاصية ColWidth لكل عمود من أعمدة الجدول إلى 1300 Twip. وكما يظهر من اسم الخاصية ColWidth فإن هذه الخاصية مسؤولة عن تحديد عرض العمود.

وهكذا تحدد grdTable.ColWidth(0) عرض العمود الأول، وتحدد grdTable.ColWidth(1) عرض العمود الثاني وهكذا...

- أضف العبارة SetColWidth (اسم الإجراء الجديد) إلى نهاية الإجراء Form_Load()، بحيث يبدو بالشكل التالي:

```
Public Sub Form_Load()
    ''''''''''''''''''''''''''''''''
    'اترك أقسام الإجراء على حالها    '
    'وأضف لها السطر الأخير           '
    ''''''''''''''''''''''''''''''''
    SetColWidth
End Sub
```

- احفظ المشروع بالطريقة المعتادة.
- نفّذ برنامج الجدول.

يُفترض أن يبدو إطار برنامج الجدول كذلك المبين بالشكل 9-6.



الشكل 9-6

توسيع عرض الأعمدة.

- استخدم مفاتيح الأسهم أو شريطي التمرير للانتقال بين الخلايا.

كما ترى من الشكل 9-7، فقد تغير عرض كل الأعمدة.

- أنهِ البرنامج بنقر الزر **خروج**.

❑ كبّر عنصر التحكم grdTable بسحب حوافه، بحيث يظهر كما في الشكل 9–8.

لاحظ أن عرض الخلايا لم يتغير، رغم تكبير كامل منطقة عنصر تحكم الجدول، السبب في ذلك أن توسيع الخلايا يتم فقط أثناء زمن التنفيذ.



الشكل 9–7

استعمال شريطي التمرير أو مفاتيح الأسهم للانتقال بين الخلايا.

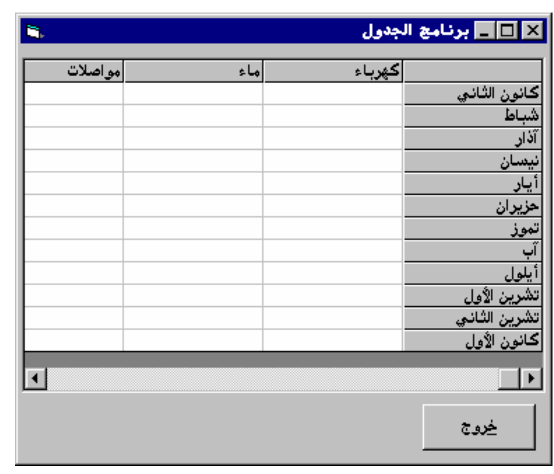


الشكل9–8

تكبير حجم عنصر الجدول أثناء زمن التصميم.

❑ نفّذ برنامج الجدول.

يظهر عنصر الجدول الآن، كذاك المبين في الشكل 9–9. كما تلاحظ المنطقة الكلية لعنصر تحكم الجدول أكبر وعرض الخلايا الآن أوسع.



الشكل 9-9
عنصر الجدول بعد
تكبيره وتوسيع خلاياه.

- تمرن على الانتقال بين الخلايا، واستخدام مفاتيح الأسهم أو شريطي التمرير.
- أنهِ برنامج الجدول بنقر الزر **خروج**.

تغيير ارتفاع الخلية

سنكتب الآن نص البرنامج المسؤول عن تغيير ارتفاع الخلايا خلال مرحلة التنفيذ.

- أضف إجراءً للنموذج frmTable بالنقر المزدوج على النموذج frmTable (لإظهار إطار نص البرنامج)، ثم اختر **Add Procedure** من القائمة **Tools** لفيجول بيسك.

يُظهر فيجول بيسك مربع الحوار Add Procedure.

- اكتب اسماً للإجراء، (ليكن SetRowHeight) في الحقل Name.
- تحقق من اختيار كل من Sub و Public ضمن مربع الحوار Add Procedure
- انقر الزر **OK**.

يُشكل فيجول بيسك إجراءً جديداً يدعى SetRowHeight، في المنطقة العامة General Area للنموذج frmTable.

- أدخل النص التالي ضمن الإجراء SetRowHeight:

```
Public Sub SetRowHeight()
    Dim Counter
    For Counter = 0 To 12
      grdTable.RowHeight(Counter) = 500
```

```
    Next
End Sub
```

تعتبر الخاصية RowHeight مسؤولة عن تحديد ارتفاع الخلية.

يَستخدم الإجراء SetRowHeight الحلقة For لتحديد ارتفاع كل سطر من أسطر عنصر الجدول بـ 500 Twips.

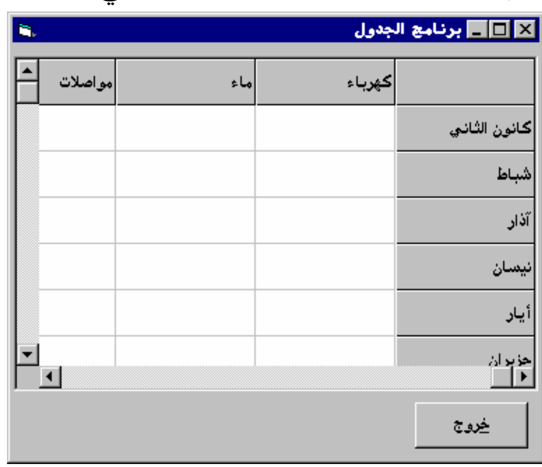
❑ أضف العبارة SetRowHeight في نهاية الإجراء Form_Load() بالشكل التالي:

```
Public Sub Form_Load()
    ''''''''''''''''''''''''''''''''
    'اترك أقسام الإجراء على حالها    '
    'و أضف لها السطر الأخير          '
    ''''''''''''''''''''''''''''''''
    SetColWidth
    SetRowHeight
End Sub
```

❑ احفظ المشروع كالعادة.

❑ نفّذ برنامج الجدول.

يُفترض أن تبدو خلايا الجدول كما في الشكل 9-10.



الشكل 9-10

زيادة ارتفاع سطور الجدول.

شريطي تمرير عنصر تحكم الجدول

لعلك لاحظت أن فيجول بيسك يضيف آلياً شريطي التمرير الأفقي والعمودي، عندما لا تتسع الخلايا ضمن المنطقة التي يحتلها عنصر الجدول. السبب في ذلك أننا تركنا قيمة الخاصية ScrollBars لعنصر الجدول على حالها، أي وفق قيمتها الافتراضية المساوية إلى 3-flexScrollBarBoth، فإذا أردت إخفاء شريطي التمرير، تستطيع تحديد القيمة 0-flexScrollBarNone للخاصية ScrollBars أثناء زمن التصميم، أو تنفيذ العبارة التالية من ضمن نص البرنامج:

```
grdTable.ScrollBars = 0
```

إذا أردت الإبقاء على شـــريط التمرير الأفقي فقط، تســتطيع إســناد القيمة 1-flexScrollBarHorizontal إلى الخاصية ScrollBars أثناء طور التصميم.

أو استخدم العبارة التالية ضمن نص البرنامج:

```
grdTable.ScrollBars = 1
```

وبالعكس إذا أردت الإبقاء على شريط التمرير العمودي فقط، تستطيع استخدام القيمة 2-flexScrollBarVertical للخاصية ScrollBars في طور التصميم.

ملاحظة

مهما تكن قيمة الخاصية ScrollBars فلا توجد مشكلة، لأنك قادر دائماً على الانتقال من خلية لأخرى بواسطة مفاتيح الأسهم من لوحة المفاتيح.

تحديد الخاصيتين Rows و Cols أثناء زمن التنفيذ

حددنا قيمتي الخاصيتين أثناء طور التصميم، فكانت Cols تساوي 5 و Rows تساوي13.

قد يكون عدد الأسطر والأعمدة معلوماً فقط، خلال زمن التنفيذ.

فمثلاً، تستطيع إعداد برنامج الجدول، ليُظهر فقط العمودين **كهرباء** و **ماء** شهرياً، (أي فاتورتي الماء والكهرباء). سيتوجب على البرنامج عندها تغيير عدد الأعمدة إلى ثلاثة فقط (واحد للعناوين اليسارية، وواحد للعنوان **كهرباء** وآخر للعنوان **ماء**).

تُستخدم عبارة مشابهة للعبارة التالية لإنجاز عملية تغيير عدد الأعمدة:

```
grdTable.Cols = 3
```

كما تُستخدم عبارة مشابهة للعبارة التالية، لتغيير عدد الأسطر:

```
grdTable.Rows = n
```

علماً أن n متحول يمثل عدد الأسطر. (إضافة لسطر عناوين الأعمدة).

ملء باقي خلايا برنامج الجدول

اتبع الخطوات التالية لملء ما تبقى من خلايا برنامج الجدول:

- أضف إجراءً جديداً إلى المنطقة General للنموذج frmTable.
- أطلق على هذا الإجراء الاسم FillCells.
- أدخل النص التالي ضمن الإجراء FillCells():

```
Public Sub FillCells()
    Dim RowCounter, ColCounter
    For ColCounter = 1 To 4
      grdTable.Col = ColCounter
      For RowCounter = 1 To 12
        grdTable.Row = RowCounter
        grdTable.Text = "غير معروف"
      Next
    Next
End Sub
```

- أضف العبارة FillCells إلى نهاية نص الإجراء Form_Load().

يفترض أن يبدو الإجراء بشكل مماثل لما يلي:

```
Public Sub Form_Load()
    ''''''''''''''''''''''''''''''''
    '    اترك أقسام الإجراء على حالها'
    '            وأضف لها السطر الأخير'
    ''''''''''''''''''''''''''''''''
    SetColWidth
```

```
    SetRowHeight
    FillCells
End Sub
```

نص الإجراء ()FillCells

لقد استخدمت حلقتين متداخلتين في الإجراء ()FillCells:

```
For ColCounter = 1 To 4
  grdTable.Col = ColCounter
  For RowCounter = 1 To 12
    grdTable.Row = RowCounter
    grdTable.Text = "غير معروف"
  Next
Next
```

ينفذ هذا الإجراء حلقتي For، وتُسنِد كلتا الحلقتين الكلمة **غير معروف** إلى الخاصية Text لكل خلية من الخلايا. تَعّد الحلقة الخارجية من 1 إلى 4، وتعد الحلقة الداخلية من 1 إلى 12.

وبذلك تتمكن الحلقتان For من تغطية كل خلايا الجدول (باستثناء السطر العلوي الذي يمثل سطر العناوين، وباستثناء العمود اليساري كذلك) وإسناد الكلمة **غير معروف** إليها.

- احفظ المشروع بالطريقة المعتادة.
- نفّذ برنامج الجدول.

تملأ خلايا عنصر التحكم بالنص غير معروف (انظر الشكل 9-11).

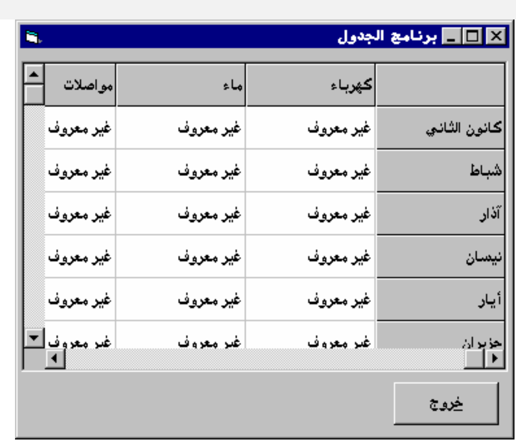
- أنهِ البرنامج بالنقر على الزر **خروج**.
- أضف الآن النص التالي إلى الإجراء ()FillCells لنملأ بعض خلايا الجدول بقيم ذات معنى:

```
Public Sub FillCells()
    Dim RowCounter, ColCounter
    For ColCounter = 1 To 4
      grdTable.Col = ColCounter
      For RowCounter = 1 To 12
        grdTable.Row = RowCounter
```

```
      grdTable.Text = "غير معروف"
    Next
  Next
  grdTable.Row = 1
  grdTable.Col = 1
  grdTable.Text = "$100.00"

  grdTable.Row = 2
  grdTable.Col = 1
  grdTable.Text = "$50.00"

  grdTable.Row = 2
  grdTable.Col = 2
  grdTable.Text = "$75.00"
End Sub
```



الشكل 9-11

ملء خلايا عنصر الجدول

بنص ما (كلمة غير معروف).

يملأ النص الذي أضفناه للتو إلى الإجراء FillCells()، ثلاث خلايا من الجدول، بإسناد رقم سطر وعمود الخلية المحددة، إلى الخاصيتين Row و Col ثم إسناد النص المطلوب إلى الخاصية Text للخلية.

- نفّذ برنامج الجدول.

كما تلاحظ، تظهر ثلاث خلايا وهي تحمل النص $100.00 و $50.00 و $75.00 (انظر الشكل 9-12).

- أضف زراً يدعى **تنظيف** إلى النموذج frmTable، حسب ما يوضحه الشكل 9-13.

- ❑ أسند له الاسم cmdClear في الخاصية Name، أما الخاصية Caption فأسند إليها العنوان &تنظيف.

يفترض أن يبدو النموذج frmTable كما في الشكل 9-13.



الشكل 9-12

ملء ثلاث خلايا

في الجدول.



الشكل 9-13

إضافة الزر **تنظيف**.

- ❑ أدخل النص التالي ضمن الإجراء cmdClear() للنموذج frmTable:

```
Private Sub cmdClear_Click()
    Dim RowCounter, ColCounter
    For ColCounter = 1 To 4
      grdTable.Col = ColCounter
      For RowCounter = 1 To 12
        grdTable.Row = RowCounter
        grdTable.Text = ""
      Next
    Next
End Sub
```

يعمل هذا الإجراء على تنظيف خلايا الجدول من المعلومات.

- احفظ المشروع باختيار **Save Project** من القائمة **File** لفيجول بيسك.
- نفّذ البرنامج.
- انقر الزر **تنظيف**.

يستجيب البرنامج بمحي كل معطيات الخلايا في عنصر تحكم الجدول.

يعمل الإجراء ()cmdClear_Click بشكل مشابه للإجراء ()FillCells باستثناء أنه يكتب لاشيء ("") بدلاً من كتابة نص في الخلايا:

```
For ColCounter = 1 To 4 Step 1
  grdTable.Col = ColCounter
  For RowCounter = 1 To 12 Step 1
    grdTable.Row = RowCounter
    grdTable.Text = ""
  Next
Next
```

وبهذا يتسبب فى حذف المعطيات السابقة. يستخدم الإجراء أيضاً حلقتين تَعّد إحداهما إلى 4 وأخرى تعد إلى 12، مما يسمح له بتغطية كل خلايا الجدول باستثناء سطر العنوان وعمود العنوان.

هناك تقنية أفضل، تتمثل باستخدام نص البرنامج التالي:

```
For ColCounter = 1 To grdTable.Cols - 1
  grdTable.Col = ColCounter
  For RowCounter = 1 To grdTable.Rows - 1
    grdTable.Row = RowCounter
    grdTable.Text = ""
  Next
Next
```

فبدلاً من تحديد النهايتين 4 و12 بشكل صريح ضمن حلقتي For، تَستخدم هذه الطريقة قيم الخاصيتين Cols وRows لعنصر تحكم الجدول. وبذلك يتم تنظيف الجدول مهما كان عدد أسطره أو أعمدته. وفي حال ما إذا عدّلت قيمة هاتين الخاصيتين أثناء طور التصميم أو أثناء زمن التنفيذ، حينها لا تحتاج لتغيير نص الإجراء.

محاذاة النص في الخلايا

تُستخدم خلايا السطر صفر (Row # 0) لإظهار عناوين الأعمدة، وتستخدم خلايا العمود الواقع أقصى اليمين لإظهار عناوين الأسطر.

تدعى هذه الخلايا بالسطور الثابتة والأعمدة الثابتة، لأنها تظل دائماً في مكانها، مهما قمت بعملية إزاحة للخلايا الأخرى. تتم إزاحة خلايا الجدول الأخرى يميناً ويساراً، وصعوداً ونزولاً بواسطة شريطي التمرير ومفاتيح الأسهم. وبالمناسبة تدعى خلايا الجدول التي ينتقل المستخدم عبرها بالخلايا غير الثابتة.

تستخدم الخاصية ColAlignment لتنظيم وضبط محاذاة الأعمدة غير الثابتة. والقيم المحتملة لهذه الخاصية مبينة في الجدول 9-1.

الجدول 9-1. الوضعيات الممكنة للبيانات ضمن خلايا الجدول.

القيمة	الشرح
0	يسار أعلى الخلية.
1	يسار وسط الخلية (قيمة افتراضية للمعلومات النصية).
2	يسار أسفل الخلية.
3	وسط أعلى الخلية.
4	مركز الخلية.
5	وسط أسفل الخلية.
6	يمين أعلى الخلية.
7	يمين وسط الخلية (قيمة افتراضية للمعلومات الرقمية).
8	يمين أسفل الخلية.
9	عام (يسار وسط للمعلومات النصية، ويمين وسط للمعلومات الرقمية).

فمثلاً لمحاذاة محتويات الخلية إلى الطرف الأيمن والأسفل من الخلية، أسند القيمة 8 إلى الخاصية ColAlignment لهذه الخلية. أما لمحاذاة محتويات خلية إلى الطرف الأيمن والأعلى من الخلية فأسند القيمة 6 إلى الخاصية ColAlignment.

استخدم الخطوات التالية لرؤية عمل الخاصية ColAlignment على أرض الواقع:

- أضف الزر **محاذاة** إلى النموذج frmTable، حسب ما يظهر في الشكل 9−14.
- أسند إلى خصائص الزر **محاذاة**، ما يلي:
  - الاسم cmdAlign إلى الخاصية Name.
  - العنوان &محاذاة إلى الخاصية Caption.



الشكل 9−14

إضافة الزر **محاذاة**.

- أدخل النص التالي ضمن الإجراء ()cmdAlign_Click للنموذج frmTable:

```
Private Sub cmdAlign_Click()
    Dim ColCounter
    For ColCounter = 1 To grdTable.Cols - 2
      grdTable.ColAlignment(ColCounter) = 4
    Next
End Sub
```

يسند هذا الإجراء القيمة 4 إلى الخاصية ColAlignment لكل الأعمدة غير الثابتة، باستثناء العمود الواقع أقصى اليسار.

وحسب ما أورده الجدول 9−1، يؤدي إسناد القيمة 4 إلى الخاصية ColAlignment لتمركز البيانات ضمن الخلية.

- نفّذ برنامج الجدول.
- انقر الزر **محاذاة**.

كما تلاحظ، يتمركز النص في كل الخلايا غير الثابتة، باستثناء الخلايا الواقعة في العمود اليساري الأخير.

السبب الذي دفعنا إلى عدم توسيط النص في العمود اليساري الأخير، تمكينك من المقارنة بين الخلايا التي يتم تغيير خاصيتها ColAlignment، وبين خلايا العمود اليساري الذي نحافظ فيه على الخاصية ColAlignment بدون تغيير (أي وفق القيمة الافتراضية).

بقي أن تعلم أننا نستطيع أيضاً ضبط (محاذاة) الأعمدة والسطور الثابتة. تُستخدم الخاصية FixedAlignment لهذا الغرض. يمكن استخدام هذه الخاصية مثلاً لمحاذاة السطور والأعمدة الثابتة بشكل مختلف عن السطور والأعمدة غير الثابتة التي تقع تحت العنوان. مثلاً، أسند القيمة 1 إلى الخاصية FixedAlignment للعمود الأول على سبيل المثال، ثم أسند أي قيمة أخرى من القيم المدرجة في الجدول 9–1 إلى الخلايا التي تقع تحت هذا العمود.

استخدم العبارة التالية، كمثال لتوسيط النص الموجود في العمود الواقع أقصى اليمين:

```
grdTable.FixedAlignment(0) = 1
```

استخدم العبارات التالية لتوسيط العناوين: **الكهرباء** و **الماء** و **الطعام** (أي توسيط عناوين الأعمدة 1 و2 و3):

```
grdTable.FixedAlignment(1) = 4
grdTable.FixedAlignment(2) = 4
grdTable.FixedAlignment(3) = 4
```

تمرن على مختلف قيم الجدول 9–1 واحكم على النتائج بنفسك.

الخلاصة

تعلمت فى هذا الفصل كيفية استخدام عنصر تحكم الجدول وكيفية ملئه بالمعلومات، أو تنظيفه من المعلومات، كما تعلمت أيضاً كيفية محاذاة المعلومات فى الخلية بشتى الأوضاع وكيف أن الجدول يحتوى على خلايا ثابتة (تستخدم كعناوين)، وخلايا متحركة (لإظهار المعلومات. يمكنك الأن تقديم المعلومات للمستخدم فى شكل جدول أنيق.

الفصل العاشر

عملية الإظهار والطباعة

سنتعلم في هذا الفصل كيفية إظهار وطباعة المعلومات، وكيفية إظهار النص بخطوط مختلفة Fonts، وكيفية صياغة الأرقام والتواريخ وإرسال المعطيات (النصية والرسومية) إلى الطابعة.

الخطوط الكتابية Fonts

هنالك نوعان من الخطوط: خطوط قابلة لتغيير حجمها (متدرجة) Scaleable، وخطوط غير قابلة لتغيير حجمها (غير متدرجة) non Scaleable.

يتشكل الخط القابل للتدرج، باستخدام الصيغ الرياضية، فمثلاً يُعرّف الحرف B لمرة واحدة فقط، وتُولّد كل الأحجام الأخرى للحرف B من نفس الحرف الأصلي بتكبيره أو تصغيره.

من جهة ثانية، يخزّن الخط غير القابل للتدرج، بشكل نقطي Bitmap، وتخزّن كذلك الأحجام الأكبر والأصغر لنفس الخط بأشكال نقطية مختلفة، بحيث يكون لكل مقاس حرف مختلف، صورة خاصة به).

ولهذا السبب، قد يظهر الخط المتدرج Scaled (المصغر خاصة) أحياناً، بجودة أقل من الخط غير القابل للتدرج non Scaleable، وخاصة على الشاشة.

استخدام خطوط مختلفة في برامجك

لا بد من اختيار نوع خط Font للنص الذي ترغب بإظهاره، وتعتبر عملية اختيار الخط المناسب عملاً هاماً، فهل سيتوافر هذا الخط لدى المستخدمين الآخرين؟.

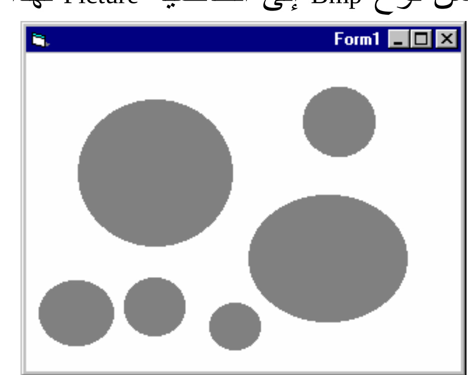
ينتقي ويندوز عند عدم توافر ذلك النوع من الخط، أقرب خط مشابه للمطلوب، ولكن عندما لا يكون لدى المستخدم خياراً واسعاً من الخطوط، فقد ينتقي ويندوز خطاً أكبر من الخط الذي يراد استخدامه، مما قد يفسد النموذج، بالتسبب بتخطي النص، بل وفي بعض الأحيان، يحاول ويندوز اختيار الخط الأشبه بالخط المطلوب، مستعيناً بخط موجود على حاسب المستخدم، مما قد يتسبب أيضاً بإفساد جمال وتناسق النماذج.

لعل الوسيلة الأسهل للتغلب على هذه المشكلة، تتمثل باستخدام الخطوط الشهيرة فقط، والتي تأتي مع برمجية النظام ويندوز الأصلية. يمكن لبرامجك أيضاً تفحص الملف Win.INI الذي يحتوي مقطعاً يحمل العنوان [Fonts]، يأتي ضمن هذا المقطع كل الخطوط المثبتة في النظام وبعد تفحص هذا المقطع يمكن للبرنامج تحديد الخط الذي سيستخدمه.

الخاصية FontTransparent

يدعم النموذج وكذلك عنصر تحكم الصورة Picture، الخاصية FontTransparent.فإذا كانت قيمة هذه الخاصية تساوي False، يظهر النص بالخلفية المشار إليها بالخاصية BackColor للكائن الذي سيتم إظهار النص عليه. فمثلاً إذا كانت BackColor للنموذج تشير إلى اللون الأزرق، فإن النص سيظهر على النموذج مع خلفية زرقاء.

يبين الشكل10-1 نموذجاً يحمل صورة نقطية ما بداخله، أي بمعنى أنه تم إسناد ملف من نوع Bmp إلى الخاصية Picture لهذا النموذج.



الشكل 10-1

النموذج frmMyForm

وبداخله صورة نقطية

من نوع BMP.

تستطيع إظهار نص داخل النموذج بواسطة الطريقة Print. سوف ينفذ الإجراء Form_Click() آلياً عند نقر النموذج:

```
Private Sub Form_Click()
    frmMyForm.FontTransparent = True
    frmMyForm.Print "هذه الجملة شفافة"
    frmMyForm.FontTransparent = False
    frmMyForm.Print "هذه الجملة غير شفافة"
End Sub
```

يُسند نص هذا الإجراء القيمة True إلى الخاصية FontTransparent للنموذج ثم يستخدم الطريقة Print:

```
    frmMyForm.Print "هذه الجملة شفافة"
```

ثم يُسند الإجراء القيمة False إلى الخاصية FontTransparent للنموذج ويستخدم الطريقة Print لإظهار النص التالي:

```
    frmMyForm.Print "هذه الجملة غير شفافة"
```

يبين الشكل 10-2 النموذج بعد النقر عليه عدة مرات (يظهر بعد كل نقرة سطري نص).

وهكذا، فالسطر الأول والثالث والخامس والسابع ... الخ، يظهر وفق القيمة True للخاصية FontTransparent، ويظهر السطر الثاني والرابع والسادس ... الخ، وفق القيمة False للخاصية FontTransparent للنموذج.

لاحظ أن السطور التي تظهر وفق القيمة True للخاصية FontTransparent تقبل اللون الذي يتم إظهارها عليه وتندمج معه، انظر للسطر الثالث مثلاً في الشكل 10-2.

بعكس السطور التي تظهر وفق القيمة False لهذه الخاصية والتي تتسبب بإخفاء جزء من الشكل الذي تظهر عليه، كأن تظهر على خلفية بيضاء (لأن الخاصية BackColor للنموذج على سبيل المثال تشير إلى اللون الأبيض).

تظهر السطور الفردية بدون أن تؤثر على الشكل، وذلك لأن قيمة الخاصية FontTransparent تساوي True، بعكس السطور الزوجية التي تساوي قيمة الخاصية FontTransparent فيها False، والتي تحذف جزءاً من الشكل.



الشكل 10-2

إظهار النص فوق الصورة المبينة في الشكل 10-1.

برنامج عرض الخطوط

يوضح برنامج عرض الخطوط مختلف خصائص الخط المتوفرة في فيجول بيسك.

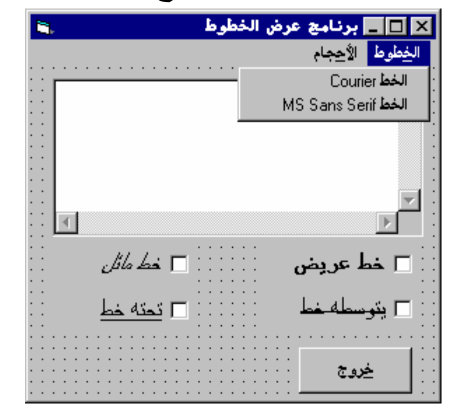
التمثيل المرئي لبرنامج عرض الخطوط

نبدأ كعادتنا بالتمثيل المرئي لنموذج البرنامج:

- أنشئ الدليل C:\VB5Prg\Ch10 لحفظ العمل فيه.

- أنشئ مشروعاً قياسياً تنفيذياً Standard EXE، واحفظ النموذج فيه بالاسم ShowFont.frm في الدليل C:\VB5Prg\Ch10 واحفظ ملف المشروع بالاسم ShowFont.VbP في الدليل C:\VB5Prg\Ch10
- أنشئ النموذج frmShowFont تبعاً للجدول 10-1.

يفترض أن يبدو النموذج المكتمل كذاك المبين في الشكل 10-3.



الشكل 10-3

النموذج frmShowFont

(طور التصميم).

الجدول 10-1. جدول خصائص النموذج frmShowFont.

الكائن	الخاصية	القيمة
Form	**Name**	**frmShowFonts**
	Caption	برنامج عرض الخطوط
	RightToLeft	True
TextBox	**Name**	**txtTest**
	Text	(اتركه فارغاً)
	MultiLine	True
	ScrollBars	3-Both
	RightToLeft	True
CommandButton	**Name**	**cmExit**
	Caption	&خروج
	RightToLeft	True

CheckBox	**Name**	**chkBold**
	Caption	خط عريض
	Font	(اجعله خطاً عريضاً)
	RightToLeft	True

الكائن	الخاصية	القيمة
CheckBox	**Name**	**chkItalic**
	Caption	خط مائل
	Font	(اجعله خطاً مائلاً)
	RightToLeft	True
CheckBox	**Name**	**chkStrike**
	Caption	يتوسطه خط
	Font	(اجعله خطاً يتوسطه خط)
	RightToLeft	True
CheckBox	**Name**	**chkUnderline**
	Caption	تحته خط
	Font	(اجعل تحته خطاً)
	RightToLeft	True
Menu	(انظر الجدول 10−2)	(انظر الجدول 10−2)

الجدول 10−2. جدول قائمة النموذج frmShowFont.

الخاصية Name	العنوان
mnuFonts	ال&خطوط
mnuCourier	...الخط Courier
mnuMSSansSerif	...الخط MS Sans Serif

mnuSize	الأ&حجام
mnu10Points	حجم 10 نقاط...
mnu12Points	حجم 12 نقطة...

إدخال نص برنامج عرض الخطوط

سنكتب الآن نص برنامج عرض الخطوط:

- أدخل النص التالي ضمن قسم التصاريح العامة:

```
' يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء chkBold_Click():

```
Private Sub chkBold_Click()
    txtTest.FontBold = chkBold.Value
End Sub
```

- أدخل النص التالي ضمن الإجراء chkItalic_Click():

```
Private Sub chkItalic_Click()
    txtTest.FontItalic = chkItalic.Value
End Sub
```

- أدخل النص التالي ضمن الإجراء chkStrike_Click():

```
Private Sub chkStrike_Click()
    txtTest.FontStrikethru = chkStrike.Value
End Sub
```

- أدخل النص التالي ضمن الإجراء chkUnderline_Click() للنموذج frmShowFont:

```
Private Sub chkUnderline_Click()
    txtTest.FontUnderline = chkUnderline.Value
End Sub
```

- أدخل النص التالي ضمن الإجراء cmdExit_Click():

```
Private Sub cmdExit_Click()
    End
End Sub
```

- أدخل النص التالي ضمن الإجراء mnu10Points_Click():

```
Private Sub mnu10Points_Click()
    txtTest.FontSize = 10
End Sub
```

- أدخل النص التالي ضمن الإجراء ()mnu12Points_Click:

```
Private Sub mnu12Points_Click()
    txtTest.FontSize = 12
End Sub
```

- أدخل النص التالي ضمن الإجراء ()mnuCourier_Click:

```
Private Sub mnuCourier_Click()
    txtTest.FontName = "Courrier"
End Sub
```

- أدخل النص التالي ضمن الإجراء ()mnuMSSansSerif_Click:

```
Private Sub mnuMSSansSerif_Click()
    txtTest.FontName = "MS Sans Serif"
End Sub
```

- احفظ المشروع بالطريقة المعتادة.

تنفيذ برنامج عرض الخطوط

لنشاهد ما كتبناه قيد التنفيذ:

- نفّذ برنامج عرض الخطوط.
- اكتب شيئاً ما داخل مربع النص (انظر الشكل 10-4).

يظهر النص الذي أدخلته بالخط الافتراضي (الخط المحدد أثناء مرحلة التصميم).



الشكل10-4
الخط الافتراضي
لبرنامج عرض الخطوط.

- ضع علامة اختيار في خانة الاختيار **خط عريض**.

يتغير خط مربع النص إلى خط غامق Bold. كما في الشكل 10-5.



الشكل10-5

وضع علامة اختيار في خانة الاختيار **خط عريض**.

- ضع علامة اختيار في خانة الاختيار **خط مائل**.

يستجيب البرنامج بتغيير نص مربع النص إلى خط مائل Italic، وبما أنه توجد علامة اختيار في خانة الاختيار ***خط عريض***، *فسيظهر النص بخط غامق ومائل في نفس الوقت (الشكل 10-6).*



الشكل 10-6

وضع علامتي اختيار في خانتي الاختيار **خط عريض** و **خط مائل**.

- ضع علامة اختيار في خانة الاختيار **يتوسطه خط** وألغها من خانتي الاختيار **خط عريض** و**خط مائل**، وراقب النتائج.

يستجيب البرنامج بإزالة الخط الغامق والخط المائل من النص، ومرور خط من النص (الشكل 10-7).



الشكل 10-7

وضع علامة اختيار

في خانة **يتوسطه خط**.

❑ ضع إشارة في خانة الاختيار **تحته خط**.

يستجيب البرنامج بتسطير النص في مربع النص (الشكل 10-8).

❑ بدّل حجم الخط باختيار حجم ما من القائمة **الأحجام**.



الشكل 10-8

وضع علامة اختيار في

خانة الاختيار **تحته خط**.

يستجيب البرنامج بتبديل حجم النص إلى الحجم الذي اخترته، يبين الشكل 10-9 مربع النص بعد اختيار الحجم 12.



الشكل 10-9

تحديد حجم الخط بإسناد

القيمة 12 للخاصية FontSize.

❑ انقر الزر **خروج** لإنهاء البرنامج.

كيف يعمل برنامج عرض الخطوط

يُغيِّر البرنامج، خصائص الخط المستخدم في مربع النص، تبعاً لاختياراتك.

نص الإجراء ()chkBlod_Click

يُنفذ هذا الإجراء آلياً، عند نقر خانة الاختيار chkBlod (خانة **خط عريض**):

```
Private Sub chkBold_Click()
    txtTest.FontBold = chkBold.Value
End Sub
```

تُسند القيمة True إلى الخاصية FontBold لمربع النص، عندما تساوي الخاصية Value لخانة الاختيار chkBold القيمة True.

وعلى العكس، تُسند القيمة False إلى الخاصية FontBold لمربع النص txtTest، عند إزالة علامة الاختيار من الخانة chkBold، أي عندما تساوي الخاصية Value لخانة الاختيار chkBold إلى False.

تعمل الإجراءات chkItalic_Click، chkStrike_Click، chkUnderline_Click بشكل مشابه لعمل الإجراء chkBold_Click.

- يُسند الإجراء ()chkItalic_Click إحدى القيمتين True أو False للخاصية FontItalic.
- يُسند الإجراء ()chkStrike_Click إحدى القيمتين True أو False للخاصية FontStrike.
- يُسند الإجراء ()chkUnderline_Click إحدى القيمتين True أو False للخاصية FontUnderline.

نص الإجراء ()mnu10Points_Click

يُنفذ الإجراء ()mnu10Points_Click عند اختيار **10نقاط** من القائمة **أحجام**:

```
Private Sub mnu10Points_Click()
    txtTest.FontSize = 10
End Sub
```

يُسند هذا الإجراء القيمة 10 إلى الخاصية FontSize لمربع النص txtText، ويعمل الإجراء mnu10Point_Click() بنفس الطريقة، ولكن يسند القيمة 12 إلى الخاصية FontSize.

نص الإجراء mnuCourier_Click()

ينفذ الإجراء mnuCourier_Click() عند اختيار **الخط Courier 10** من القائمة **أحجام:**

```
Private Sub mnuCourier_Click()
    txtTest.FontName = "Courrier"
End Sub
```

يُسند هذا الإجراء نوع الخط Courier إلى الخاصية FontName لمربع النص txtText، يعمل الإجراء mnuMsSamsSerif_Click() بطريقة مشابهة بإسناد نوع الخط MsSamsSerif للخاصية FontName.

المصطلح WYSIWYG

يقصد بالمصطلح WYSIWYG أن ما تشاهده هو ما تحصل عليه، وهو مأخوذ من العبارة What You See Is What Yow Get، ويُستخدم هذا المصطلح للدلالة على قدرة البرنامج على إنتاج نسخة طبق الأصل على الطابعة لما تشاهده على الشاشة، يتطلب الحصول على برامج WYSIWYG حقيقية مائة بالمائة، دقة برمجية كبيرة، لأن المستخدمين يمكن أن يمتلكوا طابعات وشاشات وخطوط مختلفة.

برنامج الخطوط

يوضح برنامج الخطوط كيف يمكن لبرنامجك اتخاذ القرار بشأن الخطوط المتوفرة في النظام. يمكن استعمال التقنية المستخدمة من قبل برنامج الخطوط لإنتاج برامج WYSIWYG.

التمثيل المرئي لبرنامج الخطوط

سنبدأ كعادتنا بالتمثيل المرئي للبرنامج:

- أنشئ مشروعاً تنفيذياً قياسياً Standard EXE، واحفظ نموذج المشروع بالاسم Fonts.frm في الدليل C:\VB5Prg\Ch10 واحفظ ملف المشروع بالاسم Fonts.Vbpفي الدليل C:\VB5Prg\Ch10.
- أنشئ النموذج frmFonts طبقاً للجدول 10-3.

يُفترض أن يبدو النموذج المكتمل كذاك المبين في الشكل10-10



الشكل 10-10
النموذج frmFonts
(طور التصميم).

الجدول 10-3. جدول خصائص النموذج frmFonts.

الكائن	الخاصية	القيمة
Form	**Name**	**frmFonts**
	Caption	برنامج الخطوط
	RightToLeft	True
ComboBox	**Name**	**cboFontsPrinter**
	Sorted	True
	Style	2-Dropdown List
	RightToLeft	True
ComboBox	**Name**	**cboFontsScreen**
	Sorted	True
	Style	2-Dropdown List
	RightToLeft	True
CommandButton	**Name**	**cmExit**
	Caption	&خروج
	RightToLeft	True
Label	**Name**	**lblPrinter**
	Caption	الخطوط المتاحة على الطابعة
	RightToLeft	True
Label	**Name**	**lblScreen**
	Caption	الخطوط المتاحة على الشاشة
	RightToLeft	True
Label	**Name**	**lblSample**
	Caption	فيجول بيسك5 Visual Basic5
	Alignment	2-Center
	BorderStyle	1-Fixed Single
	RightToLeft	True
Label	**Name**	**lblSampleInfo**
	Caption	مثال:
	RightToLeft	True

يطالبك الجدول 10–3 بوضع مربعي التحرير والسرد في النموذج frmFonts.

يظهر مستطيل أصفر يحمل الرسالة ComboBox (مربع التحرير والسرد) عندما تضع مؤشر الفأرة فوق رمز مربعي التحرير والسرد ComboBox في إطار مربع الأدوات، وبهذه الطريقة تتأكد من وجود هذا الرمز في مربع الأدوات.

إدخال نص برنامج الخطوط

سندخل الآن نص برنامج الخطوط:

- أدخل النص التالي ضمن قسم التصاريح العامة General Declartions للنموذج frmFonts:

```
'يجب التصريح عن كل المتحولات
Option Explicit
Dim gNumOfPrinterFonts
Dim gNumOfScreenFonts
```

- أدخل النص التالي ضمن الإجراء ()cboFontsScreen_Click:

```
Private Sub cboFontsScreen_Click()
    'اختار المستخدم خطاً جديداً على الشاشة
    'لهذا غير خط اللافتة بما يتوافق مع ما اختاره المستخدم
    lblSample.FontName = cboFontsScreen.Text
End Sub
```

- أدخل النص التالي ضمن الإجراء ()cmdNumberOfFonts_Click:

```
Private Sub cmdNumberOfFonts_Click()
    MsgBox "عدد خطوط الشاشة:" + Str(gNumOfScreenFonts)
    MsgBox "عدد خطوط الطابعة:" + Str(gNumOfPrinterFonts)
End Sub
```

- أدخل النص التالي ضمن الإجراء ()Form_Load:

```
Private Sub Form_Load()
    Dim I
    gNumOfScreenFonts = Screen.FontCount - 1
    gNumOfPrinterFonts = Printer.FontCount - 1

    For I = 0 To gNumOfScreenFonts - 1
      cboFontsScreen.AddItem Screen.Fonts(I)
```

```
    Next

    For I = 0 To gNumOfPrinterFonts - 1
      cboFontsPrinter.AddItem Printer.Fonts(I)
    Next

    cboFontsScreen.ListIndex = 0
    cboFontsPrinter.ListIndex = 0
End Sub
```

- أدخل النص التالي ضمن الإجراء cmdExit_Click():

```
Private Sub cmdExit_Click()
    End
End Sub
```

- احفظ المشروع.

تنفيذ برنامج الخطوط

لنشاهد ما كتبناه قيد التنفيذ:

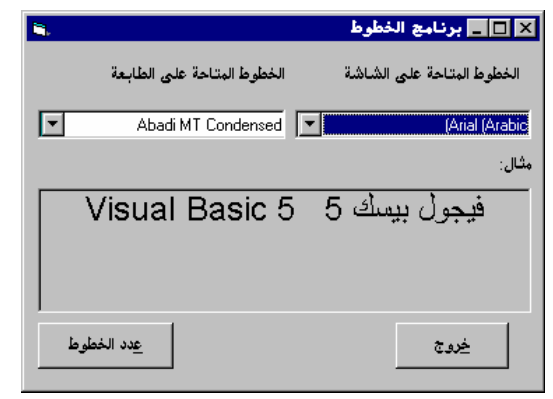
- نفّذ برنامج الخطوط.
- انقر على الزر **عدد الخطوط**.

يستجيب البرنامج بإظهار مربعي رسالة، يعطيان عدد الخطوط المتوفرة لكل من الشاشة والطابعة.

- اختر خطاً من خطوط الشاشة المتوفرة.

يتغير نوع خط اللافتة الافتراضي والمسماة **مثال**.

تبعاً للخط المختار (انظر الشكل 10-11).



الشكل 10-11

اختيار نوع الخط

.Arial Arabic

كما يوضح الشكل 10-12 ما يمكن أن يظهر عند تجول المستخدم عبر لائحة خطوط الطابعة المتوفرة.



الشكل 10-12

إظهار خطوط الطابعة.

ملاحظة

لا يتغير نوع خط النص المكتوب في اللافتة **مثال**، عند اختيار خط طابعة جديد، وإنما يتغير عند اختيار نوع خط شاشة جديد.

كيف يعمل برنامج الخطوط

يستخلص برنامج الخطوط، أنواع الخطوط المتوفرة لكل من الشاشة والطابعة ويظهرها في مربعي التحرير والسرد الخاص بكل منهما.

نص قسم التصاريح العامة

يصرح هذا القسم عن متحولين عاميين:

```
يجب التصريح عن كل المتحولات'
```

```
Option Explicit
Dim gNumOfPrinterFonts
Dim gNumOfScreenFonts
```

يمثل هذان المتحولان عدد خطوط الشاشة وخطوط الطابعة المتوفرة، تُشاهِد كل إجراءات النموذج هذين المتحولين بسبب التصريح عنهما في قسم التصاريح العامة.

نص الإجراء ()Form_Load

ينفذ هذا الإجراء عند بدء تحميل النموذج:

```
Private Sub Form_Load()
    Dim I
    gNumOfScreenFonts = Screen.FontCount - 1
    gNumOfPrinterFonts = Printer.FontCount - 1

    For I = 0 To gNumOfScreenFonts - 1
      cboFontsScreen.AddItem Screen.Fonts(I)
    Next

    For I = 0 To gNumOfPrinterFonts - 1
      cboFontsPrinter.AddItem Printer.Fonts(I)
    Next

    cboFontsScreen.ListIndex = 0
    cboFontsPrinter.ListIndex = 0
End Sub
```

يؤدي استخدام العبارة التالية إلى معرفة عدد خطوط الشاشة المتوفرة في الحاسب الشخصي الذي يُنفذ برنامج الخطوط هذا عليه، وذلك بالاستعانة بالخاصية FontCount.

```
gNumOfScreenFonts = Screen.FontCount - 1
```

وكذلك، تُسهم العبارة التالية بمعرفة عدد الخطوط المتوفرة للطابعة بالاستعانة بالخاصية FontCount.

```
gNumOfPrinterFonts = Printer.FontCount - 1
```

يملأ الإجراء بعد ذلك، مربع السرد cboFontsScreen بأنواع خطوط الشاشة:

```
For I = 0 To gNumOfScreenFonts - 1
  cboFontsScreen.AddItem Screen.Fonts(I)
Next
```

كما يملأ الإجراء مربع السرد cboFontsPrinter بأنواع خطوط الطابعة، أي:

```
For I = 0 To gNumOfPrinterFonts - 1
  cboFontsPrinter.AddItem Printer.Fonts(I)
Next
```

ملاحظة

تُستخلص الخطوط المتوفرة لكل من الطابعة والشاشة بواسطة الخاصية Fonts. فمثلاً تستخدم العبارة التالية لإسناد أول خط من خطوط الشاشة المتوفرة إلى المتحول الكتابيCurrentScreenFont :

CurrentScreenFont = Secreen.Fonts(0)

وتستخدم العبارة التالية لإسناد تاسع خط طابعة متوفر، إلى المتحول الكتابي CurrentPrinterFont: (انتبه إلى أن العد يبدأ من الصفر).

CurrentPrinterFont = Printer.Fonts(8)

ثم يُجهز الإجراء بعد ذلك مربع السرد، بحيث يشير إلى أول عنصر:

```
cboFontsScreen.ListIndex = 0
```

ثم يُبدّل خط النص الذي تورده اللافتة **مثال** (العنوان Caption) تبعاً للخاصية Text الحالية لمربع سرد أنواع خطوط الشاشة:

```
lblSample.FontName = cboFontsScreen.Text
```

وأخيراً، يُجهز الإجراء مربع السرد cboFontsPrinter بإسناد القيمة صفر للخاصية ListIndex بحيث يشير إلى أول خط متوفر:

```
cboFontPrinter.ListIndex = 0
```

نص الإجراء cboFontsScreen_Click ()

يُنفذ نص هذا الإجراء عند اختيار خط شاشة جديد من مربع السرد cboFontsScreen (وهو المربع الذي يقع تحت اللافتة "الخطوط المتاحة على الشاشة"):

```
Private Sub cboFontsScreen_click()
    lblSample.FontName = cboFontsScreen.Text
End Sub
```

يُبدّل الإجراء بعدها، خط اللافتة **مثال** إلى الخط الذي اخترته.

نص الإجراء ()cmdNumberOfFonts_Click

ينفذ هذا الإجراء تلقائياً عند نقر الزر **عدد الخطوط:**

```
Private Sub cmdNumberOfFonts_Click()
    MsgBox "عدد خطوط الشاشة:" + Str(gNumOfScreenFonts)
    MsgBox "عدد خطوط الطابعة:" + Str(gNumOfPrinterFonts)
End Sub
```

يُظهر هذا الإجراء عدد خطوط الشاشة المتوفرة، وعدد خطوط الطابعة المتوفرة أيضاً، باستخدام عبارتي MsgBox. يخزّن عدد خطوط كل من الشاشة والطابعة في المتحولين gNumOfScreenFonts و gNumOfPrinterFonts على التوالي، وتحدّث قيمة هذين المتحولين من قبل الإجراء ()Form_Load.

الطريقة Print

يمكن استخدام الطريقة Print في نموذج أو في عنصر تحكم الصورة، فمثلاً استخدم العبارة التالية لإظهار النص "Testing....." في النموذج frmMyForm:

```
frmMyForm.Print "Testing....."
```

أو استخدم العبارة التالية مثلاً، لإظهار النص ...Testing في عنصر تحكم الصورة المدعو picMyPicture:

```
picMyPicture.Print "Testing...."
```

تستخدم الفاصلة المنقوطة (;) لإخبار فيجول بيسك بوضع النص على نفس السطر، فمثلاً يؤدي استخدام العبارتين التاليتين:

```
frmMyForm.Print "This is line number 1 and ";
frmMyForm.Print "it continues..."
```

إلى توليد الخرج التالي:

```
This is line number 1 and it continues...
```

تولد العبارة التالية نفس الخرج السابق:

```
frmMyForm.Print _
    "This is line "; "number 1 and "; "it continues..."
```

محو نص

يمكنك استخدام الطريقة Cls لمحو نص كان مكتوباً في نموذج أوفي عنصر الصورة، فمثلاً لمحو النموذج frmMyForm استخدم ما يلي:

```
frmMyForm.Cls
```

أما لمحو عنصر التحكم picMyPicture فاستخدم ما يلي:

```
picMyPicture.Cls
```

تمحو الطريقة Cls النصوص والأشكال المرسومة بالطرق الرسومية المختلفة (Line، Circle، PSet ... الخ).

وضع نص عند موقع محدد

تستخدم الخاصيتان CurrentX و CurrentY لمثل هذا الغرض، فمثلاً لوضع النص Testing في النموذج frmMyForm عند العمود 5 والسطر 6، استخدم ما يلي:

```
frmForm.CurrentX = 5
frmForm.CurrentY = 6
frmMyForm.Print "Testing"
```

وبشكل مشابه، استخدم العبارات التالية لوضع النص Testing في عنصر الصورة المدعو picMyPicture عند العمود 11 والسطر 10:

```
picMyPicture.CurrentX = 11
picMyPicture.CurrentY = 10
picMyPicture.Print "Testing"
```

برنامج الفهرس

يوضح برنامج الفهرس كيف تستخدم الخاصيتين TextHeight و TextWidth اللتين تسمحان بمعرفة ارتفاع وعرض نص معين.

فمثلاً تؤدي العبارة التالية إلى إسناد قيمة ارتفاع النص "AaBbCc" إلى المتحول HeightOfabc:

```
HeightOfabc = frmMyForm.TextHeight("AaBbCc")
```

تكون القيمة المعادة من الخاصية TextHeight بنفس الوحدات المشار إليها (أي المحددة بالخاصية ScaleMode.

تصور أن النص AaBbCc عبارة عن مستطيل مغلق، تُسند العبارة السابقة ارتفاع ذلك المستطيل الوهمي إلى المتحول HeightOfabc، والذي يتحدد بارتفاع أعلى رمز بين الرموز AaBbCc.

تعتبر الخاصية TextHeight مفيدة جداً، عندما ترغب بحساب الخاصية CurrentY لسطر محدد، لنفترض مثلاً أنك أظهرت تسعة سطور، ثم رغبت بعد ذلك برسم خط أفقي تحت السطر التاسع، سنحتاج إلى حساب الإحداثيY للسطر العاشر، واستخدام هذه القيمة لإسنادها للخاصية CurrentY، إليك طريقة حساب CurrentY لمثل هذه الحالة:

```
CurrentY = frmMyForm.TextHeight("AaBbCc") * 9
```

تُعيد الخاصية TextWidth عرض النص. فمثلاً لحساب عرض النص Index، استخدم هذه العبارة:

```
WidthOfIndex = frmMyForm.TextWidth("Index")
```

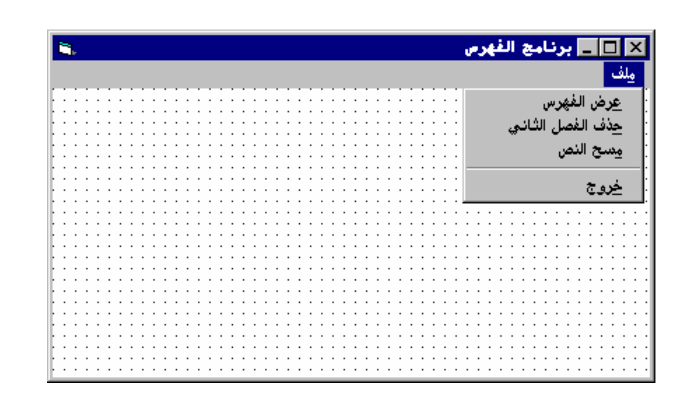
تصور ثانية، أن كلمة Index مغلقة بمستطيل، تُسند العبارة السابقة عرض ذلك المستطيل الوهمي إلى المتحول WidthOfIndex.

إذاً تعيد الخاصيتين TextHeight و TextWidth ارتفاع وعرض النص، تبعاً للقيمة الحالية للخاصية FontSize والخاصية FontName.

التمثيل المرئي لبرنامج الفهرس

- أنشئ مشروعاً قياسياً جديداً Standard EXE، واحفظ نموذج المشروع بالاسم Index.frm في الدليل C:\VB5Prg\Ch10 واحفظ ملف المشروع بالاسم Index.vbp في ذات الدليل.
- أنشئ النموذج frmIndex تبعاً للجدولين10-4 و 10-5.

يفترض أن يبدو النموذج المكتمل كذاك المبين في الشكل 10-13.



الشكل 10−13
النموذج frmIndex
(طور التصميم).

الجدول 10−4. جدول خصائص النموذج frmIndex.

القيمة	الخاصية	الكائن
frmIndex	**Name**	**Form**
برنامج الفهرس	Caption	
White	BackColor	
True	RightToLeft	
(انظر الجدول 10−5)	(انظر الجدول 10−5)	**Menu**

الجدول 10−5. جدول قائمة النموذج frmIndex.

العنوان	الخاصية **Name**
&ملف	mnuFile
...&عرض الفهرس	mnuDisplayIndex
...&حذف الفصل الثاني	mnuEraseCh2
...&مسح النص	mnuClear
...−	mnuSep1
...&خروج	mnuExit

إدخال نص برنامج الفهرس

سنباشر الآن بكتابة نص برنامج الفهرس:

❑ أدخل النص التالي ضمن قسم التصاريح العامة للنموذج frmIndex:

```
'يجب التصريح عن كل المتحولات
Option Explicit
Dim gDots
```

❑ أدخل النص التالي ضمن الإجراء Form_Load() للنموذج frmIndex:

```
Private Sub Form_Load()
    gDots = String$(84, ".")
End Sub
```

❑ أدخل النص التالي ضمن الإجراء mnuClear_Click() للنموذج frmIndex:

```
Private Sub mnuClear_Click()
    frmIndex.Cls
End Sub
```

❑أدخل النص التالي ضمن الإجراء mnuDisplayIndex_Click() للنموذج frmIndex:

```
Private Sub mnuDisplayIndex_Click()
    frmIndex.Cls
    CurrentY = 100
    CurrentX = (frmIndex.ScaleWidth - _
              frmIndex.TextWidth("الفهرس ")) / 2
    frmIndex.FontUnderline = True
    frmIndex.Print "الفهرس"

    frmIndex.FontUnderline = False
    CurrentY = frmIndex.TextHeight("VVV") * 2
    CurrentX = 100
    Print "الفصل 1" + gDots + "الكمبيوتر"
    CurrentY = frmIndex.TextHeight("VVV") * 3
    CurrentX = 100
    Print "الفصل 2" + gDots + "الشاشة"
    CurrentY = frmIndex.TextHeight("VVV") * 4
    CurrentX = 100
    Print "الفصل 3" + gDots + "الطابعة"
    CurrentY = frmIndex.TextHeight("VVV") * 5
    CurrentX = 100
    Print "الفصل 4" + gDots + "النهاية"
```

```
End Sub
```

❑ أدخل النص التالي ضمن الإجراء ()mnuEraseCh2_Click للنموذج frmIndex:

```
Private Sub mnuEraseCh2_Click()
    Dim LengthOfLine
    Dim HeightOfLine
    CurrentY = frmIndex.TextHeight("VVV") * 3
    CurrentX = ScaleWidth - 100
    LengthOfLine = frmIndex.TextWidth("الفصل 2" + _
                    gDots + "الشاشة")
    HeightOfLine = frmIndex.TextHeight("ل")
    frmIndex.Line -Step(-LengthOfLine, HeightOfLine), _
                    RGB(255, 255, 255), BF
End Sub
```

❑ أدخل النص التالي ضمن الإجراء ()mnuExit_Click للنموذج frmIndex:

```
Private Sub mnuExit_Click()
    End
End Sub
```

❑ احفظ المشروع بالطريقة المعتادة.

تنفيذ برنامج الفهرس

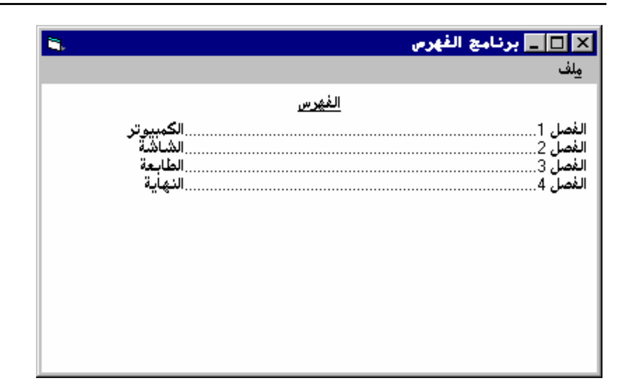
❑ نفّذ برنامج الفهرس.

❑ انقر **عرض الفهرس** من القائمة **ملف**.

يستجيب البرنامج بإظهار الفهرس المبين في الشكل 10-14.

❑ انقر **حذف الفصل الثاني** من القائمة **ملف**.

يستجيب البرنامج بحذف سطر الفصل الثاني (انظر الشكل 10-15).



الشكل 10-14

إظهار الفهرس.



الشكل 10-15

حذف سطر

الفصل الثاني.

- اختر **مسح النص** من القائمة **ملف**.

يستجيب البرنامج بحذف النص من النموذج.

- انقر زر **الخروج** لإنهاء البرنامج.

كيف يعمل برنامج الفهرس

يُظهر برنامج الفهرس نصاً باستخدام الطريقة Print.كما تستطيع إظهار النص في أي موقع بتحديد أو تعديل قيمة الخاصتين CurrentX، CurrentY.

نص قسم التصاريح العامة

يُصرح عن المتحول gDots في هذا القسم مما يعني أنه يصبح مرئياً لكل إجراءات النموذج frmIndex.

نص الإجراء Form_Load()

يُنفذ هذا الإجراء آلياً عند تحميل النموذج frmIndex:

```
Private Sub Form_Load()
    gDots = String$(84, ".")
End Sub
```

يحفظ هذا الإجراء 84 نقطة في المتحول gDots.

نص الإجراء mnuClear_Click

يُنفذ هذا الإجراء آلياً عند اختيار **مسح النص** من القائمة **ملف**:

```
Private Sub mnuClear_Click()
    frmIndex.Cls
End Sub
```

يستخدم هذا الإجراء الطريقة Cls لمحوالنموذج.

نص الإجراء mnuDisplayIndex_Click

ينفذ الإجراء mnuDisplayIndex_Click عند اختيار **عرض الفهرس** من القائمة **ملف**. يبدأ الإجراء بمحو النموذج لحذف أي نص سابق ثم يسند القيمة 100 إلى الخاصية CurrentY:

```
    frmIndex.Cls
    CurrentY = 100
```

وبما أن CurrentY تساوي الآن 100، فهذا يعني أن النص سوف يظهر على بعد 100 Twips عن قمة النموذج كما استخدمت العبارة التالية:

```
    CurrentX = (frmIndex.ScaleWidth - _
             frmIndex.TextWidth(2 / "الفهرس")
```

لإظهار النص Text عند منتصف النموذج (ارجع إلى الشكل 10−15).

يلي ذلك إسناد القيمة True إلى الخاصية FontUnderLine للنموذج، ثم استخدام الطريقة Print لطباعة كلمة الفهرس:

```
    frmIndex.FontUnderline = True
    frmIndex.Print "الفهرس"
```

ثم يعيد الإجراء إسناد القيمة False إلى الخاصية FontUnderLine لإظهار ما تبقى من نصوص بدون تسطير:

```
    frmIndex.FontUnderline = False
```

تُحدّث قيمة CurrentY لسطر "الفصل 1" كما يلي:

```
CurrentY = frmIndex.TextHeight("VVV") * 2
```

طبعاً استخدمت الأحرف "VVV" لمعرفة ارتفاع النص، بفرض أن هذه الأحرف هي أسمك الأحرف الموجودة.

وأُسندت القيمة 100 إلى CurrentX بحيث يبعد السطر بمقدار 100Twips عن يمين النموذج:

```
CurrentX = 100
```

والآن، وبعد تحديث قيمة كل من CurrentX وCurrentY، استخدمت الطريقة Print لإظهار النص:

```
Print "الفصل 1" + gDots + "الكمبيوتر"
```

يتم إظهار ما تبقى من أسطر بنفس الطريقة.

يؤدي تجاهل كتابة اسم الكائن قبل الطريقة Print إلى إظهار النص على النموذج الفعال حالياً وهكذا، فإن العبارة:

```
frmIndex.Print "الفهرس"
```

تولد نفس النتيجة التي تنتجها العبارة:

```
Print "الفهرس"
```

كذلك تُحدّث قيمة الخاصيتين CurrentX وCurrentY في هذا الإجراء، دون ذكر اسم النموذج قبلهما، لأن فيجول بيسك يفترض النموذج الفعال الحالي عند عدم ذكر اسم الكائن، فمثلاً، بفرض أن النموذج الحالي الفعال هو frmIndex، تعتبر العبارتان التاليتان متطابقتين:

```
frmIndex.CurrentY = 100
CurrentY = 100
```

نص الإجراء ()mnuEraseCh2_Click

يُنفذ هذا الإجراء عند اختيار **حذف الفصل الثاني** من القائمة **ملف**:

```
Private Sub mnuEraseCh2_Click()
    Dim LengthOfLine
    Dim HeightOfLine
    CurrentY = frmIndex.TextHeight("VVV") * 3
```

```
    CurrentX = ScaleWidth - 100
    LengthOfLine = frmIndex.TextWidth("2 الفصل" + _
                    gDots + "الشاشة")
    HeightOfLine = frmIndex.TextHeight("ل")
    frmIndex.Line -Step(-LengthOfLine, HeightOfLine), _
                    RGB(255, 255, 255), BF
End Sub
```

يسند الإجراء، الموقع الذي سيبدأ منه سطر "الفصل 2"إلى الخاصيتين CurrentX وCurrentY ويرسم مربعاً ذا خلفية بيضاء، ويحسب عرض المربع باستخدام الخاصية TextWidth.

إظهار الجداول

تستطيع استخدام الطريقة Print لإظهار جداول في النموذج أو في عنصر تحكم الصورة.

اتبع الخطوات التالية لرؤية كيفية إنجاز ذلك:

- أضف القائمة **جدول** وبند القائمة **عرض جدول** إلى نظام قوائم برنامج الفهرس.

الجدول 10-6. جدول قوائم النموذج frmIndex.

العنوان	الخاصية Name
&ملف	mnuFile
&عرض الفهرس...	mnuDisplayIndex
&حذف الفصل الثاني...	mnuEraseCh2
&مسح النص...	mnuClear

...–	mnuSep1
...&خروج	mnuExit
&جدول	mnuTable
...ع&رض الجدول	mnuDisplayTable

أدخل النص التالي ضمن الإجراء mnuDisplayTable_Click للنموذج frmIndex:

```
Private Sub mnuDisplayTable_Click()
    frmIndex.Cls
    frmIndex.FontName = "MS Sans Serif"
    frmIndex.FontSize = 10
    frmIndex.Print "الفصل" , "الشرح" , "الصفحة"
    frmIndex.Print
    frmIndex.Print "1" , "الكمبيوتر" , "1"
    frmIndex.Print "2" , "الشاشة" , "14"
    frmIndex.Print "3" , "الطابعة" , "45"
    frmIndex.Print "4" , "النهاية" , "65"
End Sub
```

- احفظ المشروع بالطريقة المعتادة.

تنفيذ نسخة برنامج الفهرس المحسنة

لنشاهد ما كتبناه قيد التنفيذ:

- نفّذ النسخة المحسنة من برنامج الفهرس.
- اختر **عرض جدول** من القائمة **جدول**.

يستجيب البرنامج بإظهار الجدول المبين في الشكل 10-16.

- اختر **خروج** من القائمة **ملف** لإنهاء البرنامج.

الشكل 10-16

إظهار فهرس جدولي.

يستخدم نص الإجراء mnuDisplayTable_Click الطريقة Print مع الفواصل (,):

```
frmIndex.Print "الفصل" , "الشرح" , "الصفحة"
```

تتسبب هذه العبارة بإظهار السلسلة الأولى بدءاً من العمود صفر، وتظهر السلسلة الثانية بدءاً من العمود 14، أما السلسلة الثالثة (أي "الصفحة") فتظهر ابتداءً من العمود 28.

يفهم فيجول بيسك من الفواصل، تغيير مواقع الطباعة إلى مناطق يحدد فيجول بيسك كل منطقة طباعة بـــ 13 رمز في الحالة الافتراضية.

ملاحظة

استخدم طريقة Print مع الفواصل لطباعة السلاسل التي تبدأ عند مواقع طباعة مختلفة فمثلاً، تؤدي العبارات التالية:

```
Print "abc","def","ghi"
Print "nop","qrs","tuf"
Print "abc","def","ghi"
```

إلى توليد الخرج التالي:

```
abc     def     ghi
nop     qrs     tuf
abc     def     ghi
```

تعريف مناطق جديدة للطباعة

يمكنك استخدام التابع الوظيفي Tab() لتعريف مناطق جديدة للطباعة.

استبدل نص الإجراء mnuDisplayTable_Click() بالنص التالي:

```
Private Sub mnuDisplayTable_Click()
    frmIndex.Cls
    frmIndex.FontName = "MS Sans Serif"
    frmIndex.FontSize = 10
    frmIndex.Print Tab(5); "الفصل"; Tab(20); _
              "الشرح"; Tab(50); "الصفحة"
    frmIndex.Print
    frmIndex.Print Tab(5); "1"; Tab(20); _
               "الكمبيوتر"; Tab(50); "1"
    frmIndex.Print Tab(5); "2"; Tab(20); _
               "الشاشة"; Tab(50); "14"
```

```
    frmIndex.Print Tab(5); "3"; Tab(20); _
                  "الـطـابـعـة"; Tab(50); "45"
    frmIndex.Print Tab(5); "4"; Tab(20); _
                  "الـنـهـايـة"; Tab(50); "65"
End Sub
```

- نفّذ برنامج الفهرس.
- اختر **عرض جدول** من القائمة **جدول**.

يستجيب البرنامج بإظهار الجدول كما في الشكل 10-17.

الشكل 10-17

إطار برنامج الفهرس.

يحدد وسيط التابع الوظيفي Tab() رقم العمود الذي سيظهر النص عنده، فمثلاً تظهر العبارة التالية النص بدءاً من العمود الخامس:

```
    frmIndex.Print Tab(5); "1"; Tab(20); _
                  "الـكـمـبـيـوتـر"; Tab(50); "1"
```

الوسيط الأول في هذه العبارة هو Tab(5)، والذي يُسند القيمة 5 للخاصية CurrentX، أما الفاصلة المنقوطة فتعني لفيجول بيسك، استئناف الطباعة على نفس السطر الحالي.

الوسيط الثالث في العبارة هو Tab(20)، والذي يُسند القيمة 20 للخاصية CurrentX، وتأمر الفاصلة المنقوطة فيجول بيسك، بإبقاء الطباعة على نفس السطر وبهذا تظهر السلسلة "الكمبيوتر" بدءاً من العمود 20.

إذاً، يمكّنك التابع الوظيفي Tab() من إظهار النص عند أي عمود تشاء.

صياغة الأرقام والتواريخ والأزمنة

يُستخدم التابع الوظيفي Format$() لإظهار الأرقام والتواريخ والأزمنة بطرق مختلفة. توضح الفقرتان التاليتان كيف يعمل هذا التابع.

صياغة الأرقام

يمكنك التحكم بالطريقة التي يُظهر فيجول بيسك الأرقام بها، وذلك بواسطة التابع Format$(). يمتلك هذا التابع وسيطين: الأول يمثل الرقم المطلوب إظهاره، والثاني يخدم كتعليمات صياغة Format Instruction، فمثلاً لإظهار الرقم 45.6 بحيث يسبقه عدد من الأصفار، استخدم العبارة التالية:

```
Print Format$(45.6,"000000.00")
```

فيتولد ما يلي:

```
000045.60
```

الرقم 45.6 يمتلك خانتين على يسار الفاصلة العشرية وخانة واحدة على يمينها، أما الوسيط الثاني فيحتوي (000000.00)، مما يعني وجوب إظهار ست خانات على يسار الفاصلة العشرية وخانتين على يمينها، وبما أن 45.6 يمتلك خانتين فقط على يسار الفاصلة العشرية، يضيف فيجول بيسك أربعة أصفار قبل الرقم، وبما أن 45.6 يمتلك خانة واحدة على يمين الفاصلة العشرية، يضيف فيجول بيسك صفر واحد في نهاية الرقم.

تستخدم هذه الميزة، لإظهار الأعداد في عمود، عندما تدعو الحاجة لإظهار الفواصل العشرية تحت بعضها، أي كما يلي:

```
000324.45
000123.40
123456.67
000004.90
132123.76
```

صياغة التواريخ والأزمنة

تستطيع مثلاً، استخدام العبارة التالية لإظهار تاريخ اليوم الحالي:

```
Print Format$(Now,"m/d/yy")
```

فإذا كان اليوم الحالي هو 4 تموز 1998، فسوف ينتج الخرج التالي:

```
7/4/98
```

يستخدم التابع الوظيفي Now، كأول وسيط في التابع Format$() للحصول على التاريخ، ويزود الوسيط (m/d/yy) طريقة لصياغة التاريخ بشكل شهر/يوم/سنة.

لاحظ أن خرج التابع Now يتعلق بقيمتي تاريخ ووقت الحاسب PC المستخدم.

يمكنك إظهار التاريخ بصيغ أخرى فمثلاً استخدم العبارة التالية:

```
Print Format$(Now,"dddd, mmmm dd, yyyy")
```

لتوليد الخرج التالي:

```
Sunday, April 11, 1999
```

واستخدم العبارة التالية:

```
Print Format$(Now,"mmmm-yy")
```

لتوليد الخرج التالي:

```
April-99
```

يمكن استخدام التابع الوظيفي Now أيضاً لإظهار الوقت الحالي. فمثلاً تستطيع استخدام هذه العبارة:

```
Print Format$(Now,"h:mm:ss a/p")
```

لتوليد الخرج التالي:

```
4:23:59 a
```

برنامج الطباعة

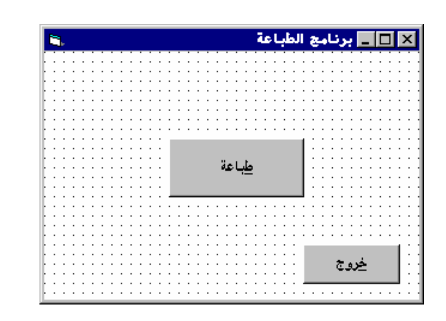
يستعرض برنامج الطباعة مدى سهولة إرسال المعطيات إلى الطابعة، بواسطة الطريقة PrintForm.

التمثيل المرئي لبرنامج الطباعة

سنبدأ كعادتنا بالتمثيل المرئي لنموذج البرنامج:

- أنشئ مشروعاً قياسياً تنفيذياً جديداً Standard EXE، واحفظ نموذج المشروع بالاسم Print.Frm في الدليل C:\VB5Prg\Ch10 واحفظ ملف المشروع بالاسم Print.Vbp في ذات الدليل.
- أنشئ النموذج frmPrint وفق الجدول 10-7.

يفترض أن يبدو النموذج المكتمل كذاك المبين في الشكل 10-18.



الشكل 10-18
النموذج frmPrint
(طور التصميم).

الجدول 10-7. جدول خصائص النموذج frmPrint.

القيمة	الخاصية	الكائن
frmPrint	**Name**	**Form**
برنامج الطباعة	Caption	
Whit	BackColor	
True	RightToLeft	
cmPrint	**Name**	**CommandButton**
&طباعة	Caption	
True	RightToLeft	
cmExit	**Name**	**CommandButton**
&خروج	Caption	
True	RightToLeft	

إدخال نص برنامج الطباعة

- تأكد من احتواء قسم التصاريح العامة للنموذج frmPrint على العبارة Option Explicit:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء cmdExit_Click() للنموذج frmPrint:

```
Private Sub cmdExit_Click()
    End
End Sub
```

- أدخل النص التالي ضمن الإجراء cmdPrint_Click() للنموذج frmPrint:

```
Private Sub cmdPrint_Click()
    Printer.Print "بسم الله الرحمن الرحيم"
    Printer.EndDoc
```

```
End Sub
```

- احفظ المشروع باختيار **Save Project** من القائمة **File** لفيجول بيسك.

تنفيذ برنامج الطباعة

- تحقق بأن الطابعة جاهزة للطباعة.
- نفّذ برنامج الطباعة.
- انقر على الزر **طباعة**.

يستجيب البرنامج باستخدام الطابعة لطباعة هذا النص:

```
"بسم الله الرحمن الرحيم"
```

كيف يعمل برنامج الطباعة

يستخدم البرنامج الطريقة Print لإرسال المعطيات إلى الطابعة. أما الطريقة EndDoc فترسل أمر بدء الطباعة إلى جهاز الطابعة.

نص الإجراء cmdPrint_Click()

ينفذ الإجراء cmdPrint_Click() عند النقر على زر **طباعة:**

```
Private Sub cmdPrint_Click()
    Printer.Print "بسم الله الرحمن الرحيم"
    Printer.EndDoc
End Sub
```

ويستخدم الطريقة Print مع الكائن Printer. تتم طباعة الوسيط الممرر للطريقة Print (أي "بسم الله الرحمن الرحيم")، حال تنفيذ الطريقة EndDoc.

ملاحظة

تستطيع الذهاب للفصل 22، وقراءة الفقرة الخاصة بتعريب كائن الطابعة، لزيادة الفائدة.

تحسين برنامج الطباعة

تُرسل الطريقة Print النص إلى الطابعة، سنحسن الآن برنامج الطباعة بحيث يطبع محتويات النموذج frmPrint:

- استبدل نص الإجراء cmdPrint_Click() بالنص التالي:

```
Private Sub cmdPrint_Click()
    frmPrint.PrintFrom
    Printer.EndDoc
End Sub
```

- احفظ المشروع بالطريقة المعتادة.

تنفيذ النسخة المحسنة لبرنامج الطباعة

لنشاهد أثر ما كتبناه:

- نفّذ نسخة برنامج الطباعة المحسنة.

يستجيب البرنامج بطباعة محتويات النموذج.

- انقر الزر **خروج** لإنهاء البرنامج.

نص نسخة الإجراء cmdPrint_Click() المحسنة

تُرسل الطريقة PrintForm محتويات النموذج إلى الطابعة نقطة نقطة (Pixel By Pixel):

```
frmPrint.PrintFrom
```

وآخر شيء يفعله هذا الإجراء هو تنفيذ الطريقة EndDoc التي تحمل الطابعة على طباعة المعطيات التي تلقتها.

ملاحظة

تستخدم الطريقة PrintForm لإرسال صورة عن النموذج إلى الطابعة، فمثلاً لإرسال محتويات النموذج frmMyForm إلى الطابعة، استخدم العبارة التالية:

frmMyForm.PrintForm

كان مفروضاً في الإصدارات السابقة لفيجول بيسك إسناد القيمة True إلى الخاصية AutoRedraw للنموذج المطلوب طباعته، حتى تتمكن الخاصية PrintForm من

العمل بشكل مناسب. أما في إصدار فيجول بيسك 5، فلا داعي لذلك.

طباعة عدة صفحات

يمكنك طباعة عدة صفحات باستخدام الطريقة NewPage. يحتفظ فيجول بيسك بمسار عدد الصفحات المطبوعة، عن طريق تحديث الخاصية Page.

اتبع الخطوات التالية لطباعة عدة صفحات:

- استبدل نص الإجراء cmdPrint_Click() بالنص التالي:

```
Private Sub cmdPrint_Click()
    Printer.Print "This is Page Number " + Str(Printer.Page)
    Printer.NewPage
    Printer.Print "This is Page Number " + Str(Printer.Page)
    Printer.EndDoc
End Sub
```

- نفّذ برنامج الطباعة.
- انقر زر **طباعة**.

يستجيب البرنامج بطباعة صفحتين:

- *الصفحة الأولى تحمل الرسالة This is Page Number 1.*
- *الثانية تحمل الرسالة This is Page Number 2.*

- انقر الزر **خروج** لإنهاء البرنامج.

يطبع الإجراء cmdPrint_Click() السلسلة التالية:

```
Printer.Print "This is Page Number " + Str(Printer.Page)
```

تحتوي الخاصية Page رقم الصفحة الحالية. يحدّث فيجول بيسك قيمة الخاصية Page بإسناد القيمة 1 إليها باعتبار أن هذه الصفحة هي أول صفحة مطبوعة.

يصرح الإجراء بعد ذلك عن صفحة جديدة باستخدام الطريقة NewPage:

```
Printer.NewPage
```

ويرسل هذه السلسلة إلى الطابعة:

```
Printer.Print "This is Page Number " + Str(Printer.Page)
```

وبما أن القيمة الحالية لـــ Page تساوي 2، تطبع الطابعة النص:

```
This is Page Number 2
```

طباعة الصور والرسوم

إذا كان المطلوب طباعة صورة ما، فيمكن وضعها في عناصر تحكم الصور Picture Controls، ثم طباعة النموذج باستخدام الطريقة PrintForm، بمعنى آخر، بغض النظر عن ما تضعه في النموذج، فإنه سوف يرسل للطابعة.

الطباعة بجودة أفضل

يوضح برنامج الطباعة، مدى سهولة إرسال المعطيات إلى الطابعة بواسطة إحدى الطريقتين Print أوPrintForm، وللحصول على دقة أفضل تستطيع استخدام الطرق الرسومية الأخرى Pest و Line و Circle الموصوفة سابقاً.

الرسم على الطابعة يتم كالرسم على النموذج، فتستطيع تحديد الخاصيتين CurrentX و CurrentY كما في العبارتين:

```
Printer.CurrentX = 0
Printer.CurrentY = 0
```

واستخدام خصائص مثل:

```
Printer.ScaleLeft
Printer.ScaleTop
Printer.Width
Printer.Height
```

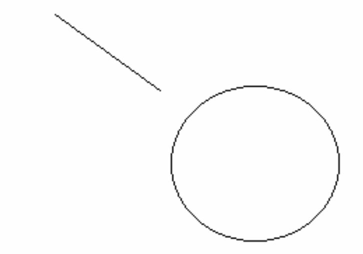
أو تستطيع استخدام الخاصيتين TextHight و TextWidth لوضع النص في مكان محدد على الورقة.

- استبدل نص الإجراء cmdPrint_Click() بالنص التالي:

```
Private Sub cmdPrint_Click()
    Printer.DrawWidth = 4
    Printer.Line (1000, 1000)-Step(1000, 1000)
    Printer.Cirle(3000, 3000), 1000
    Printer.EndDoc
End Sub
```

يُسند هذا الإجراء القيمة 4 إلى الخاصية DrawWidth للطابعة، ثم يرسم خطاً ودائرة، وينفذ الطريقة EndDoc. يبين الشكل 10-19 الخرج الناتج.

كما تلاحظ لا يوجد اختلاف بين نص البرنامج الذي يرسم على نموذج وبين الرسم على الطابعة.



الشكل 10-19
رسم خط ودائرة
على الطابعة.

الخلاصة

تعلمنا فى هذا الفصل كيفية تحديد الخط المستعمل في النصوص بواسطة الخصائص FontName و FontSize و FontItalic و FontUnderLine و FontTransparent.

كما تعلمنا كيف نستخلص الخطوط المتوفرة لكل من الشاشة والطابعة بواسطة الخاصية Fonts، وكيف نستخلص عدد هذه الخطوط بواسطة الخاصية FontCount وكيفية استخدام الخاصيتين TextWidth و TextHight إضافة إلى استعمال التابع الوظيفي ()Tab.

كما تناول هذا الفصل آلية إرسال المعطيات DATA (نصوص أو رسوم) إلى الطابعة باستخدام تقنيتين:

- الطريقة PrintForm التي ترسل صورة النموذج نقطة نقطة إلى الطابعة.
- الطريقة Print والطرق الرسومية، مثل:

```
Printer.Print "Abc"
Printer.Line -(1000, 1000)
Printer.Cirle(400, 500), 800
```

الفصل الحادي عشر

التفاعل بين تطبيقات الويندوز

بما أن البرامج التي تكتبها في فيجول بيسك، هي تطبيقات ويندوز. فإن هذا يعني أن بإمكان تطبيقاتك الاستفادة من ميزات نظام الويندوز. سوف تتعلم في هذا الفصل استخدام ميزتين: الحافظة وحلقات زمن التوقف.

الحافظة

يمكن لبرامج فيجول بيسك استخدام حافظة النظام ويندوز. والحافظة عبارة عن منطقة تُستخدم لنقل المعطيات، سواء أكانت نصوصاً أم صوراً.

برنامج الحافظة

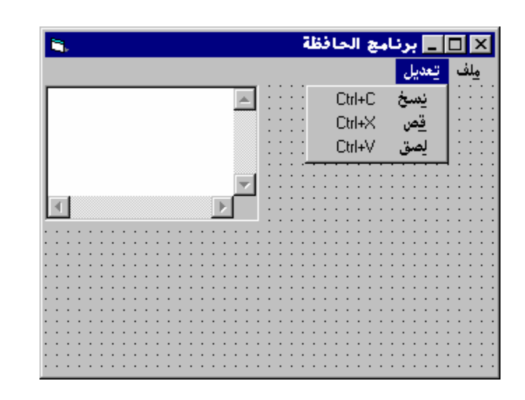
سنكتب الآن برنامجاً يدعى برنامج الحافظة، يمكّنك من كتابة أي نص في مربع نص. وإنجاز عمليات التحرير القياسية (نسخ، قص، لصق) في ويندوز. يمتلك البرنامج قائمة تعديل قياسية. يمكن استخدامها لنسخ Copy أو قص Cut أو لصق Paste أي نص.

التمثيل المرئي لبرنامج الحافظة

سنبدأ كعادتنا بالتمثيل المرئي لنموذج البرنامج.

- أنشئ الدليل C:\VB5Prg\Ch11، لحفظ العمل المنجز في هذا الدليل.
- أنشئ مشروعاً جديداً من النوع Standard EXE.
- احفظ نموذج المشروع باسم Clip.Frm في الدليل السابق واحفظ ملف المشروع باسم Clip.Vbp في ذات الدليل.
- أنشئ النموذج frmClip طبقاً للجدولين 11-1 و 11-2.

يفترض أن يبدو النموذج المكتمل كذلك المبين في الشكل 11-1.



الشكل 11-1

النموذج frmClip.

الجدول 11-1. جدول خصائص برنامج الحافظة.

الكائن	الخاصية	القيمة
Form	**Name**	**frmClip**
	RightToLeft	True
	Caption	برنامج الحافظة
Text Box	**Name**	**TxtUserArea**
	MultiLine	True
	ScrollBars	3-Both
	Text	(اجعله فارغاً)
	Left	0
	Top	0
Menu	انظر الجدول 11-2	انظر الجدول 11-2

الجدول 11-2. جدول قائمة برنامج الحافظة.

العنوان	الاسم	مفتاح الاختزال Shortcut
&ملف	mnuFile	None
...&خروج	mnuExit	None
&تعديل	mnuEdit	None
...&نسخ	mnuCopy	Ctrl+C
...&قص	mnuCut	Ctrl+X

Ctrl+V	mnuPaste	&لصق...

إدخال نص برنامج الحافظة

سنباشر الآن كتابة نص برنامج الحافظة:

- تحقق بأن قسم التصاريح العامة يحوي العبارة:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء Form_Resize():

```
Private Sub Form_Resize()
    txtUserArea.Width = frmClip.ScaleWidth
    txtUserArea.Height = frmClip.ScaleHeight
End Sub
```

- أدخل النص التالي ضمن الإجراء munCopy_Click():

```
Private Sub mnuCopy_Click()
    Clipboard.Clear
    Clipboard.SetText txtUserArea.SelText
End Sub
```

- أدخل النص التالي ضمن الإجراء munCut_Click():

```
Private Sub mnuCut_Click()
    Clipboard.Clear
    Clipboard.SetText txtUserArea.SelText
    txtUserArea.SelText = ""
End Sub
```

- أدخل النص التالي ضمن الإجراء munPaste_Click():

```
Private Sub mnuPaste_Click()
    txtUserArea.SelText = Clipboard.GetText()
End Sub
```

- أدخل النص التالي ضمن الإجراء munExit_Click():

```
Private Sub mnuExit_Click()
    End
End Sub
```

- احفظ المشروع باختيار البند **Save Project** من قائمة **File** لفيجول بيسك.

تنفيذ برنامج الحافظة

لنشاهد أثر ما كتبناه:

- نفّذ برنامج الحافظة ولاحظ مزاياه الكثيرة (رغم أننا لم نكتب إلا قدراً يسيراً من النص).
- تمرن على عمليات القص والنسخ واللصق التي يوفرها هذا البرنامج، جرب مثلاً كتابة نص في مربع النص ثم انسخ جزء من هذا النص إلى مكان آخر ضمن مربع النص ذاته أو خارجه كأن تنقله إلى برنامج وورد Word أو إلى الدفتر Word Pad.
- اختر البند **خروج** من قائمة **ملف** لإنهاء البرنامج.

كيف يعمل برنامج الحافظة

يستخدم برنامج الحافظـــة الإجراء Form_Resize() لملء كامل النموذج بمربع النص. وتنجز الإجراءات munCopy_Click() و munCut_Click() و munPaste_Click() عمليات التحرير القياسية (قص ونسخ ولصق).

الإجراء Form_Resize()

ينفذ هذا الإجراء آلياً عند ظهور النموذج لأول مرة (عند تشغيل البرنامج) أو عند تغيير حجم النموذج.

يفترض أن يملأ مربع النص كامل منطقة العمل في النموذج أثناء التنفيذ. ولهذا يعتبر الإجراء Form_Resize() مكاناً مناسباً لوضع جزء البرنامج الذي يمكن مربع النص من ملء النموذج، وتستطيع إنجاز ذلك بإسناد قيمة الخاصيتين ScaleWidth و ScaleHeight للخاصيتين Width و Height التابعتين لمربع النص على التوالي:

```
Private Sub Form_Resize()
    txtUserArea.Width = frmClip.ScaleWidth
    txtUserArea.Height = frmClip.ScaleHeight
End Sub
```

لنفترض أنك أسندت للخاصيتين Top و Left التابعتين لمربع النــص القيــمة صفر أثناء مرحلة التصميم، بما معناه أن الزاوية اليسرى العليا لمربع النص يجب أن تقع عند الزاوية اليسرى العليا للنموذج. وعندما ينفذ الإجراء Form_Resize() يحدد عرض مربع النص بعرض النموذج ScaleWidth وارتفاعه بارتفاع النموذج ScaleHeight.

نص الإجراء munCopy_Click()

ينفذ الإجراء munCopy_Click() عند اختيار البند **نسخ** من قائمة **تعديل:**

```
Private Sub mnuCopy_Click()
    Clipboard.Clear
    Clipboard.SetText txtUserArea.SelText
End Sub
```

ويعمل على نسخ الجزء المضاء من النص إلى الحافظة.

تمحو أول عبارة في الإجراء محتويات الحافظة، أي:

```
    Clipboard.Clear
```

ثم تنسخ المنطقة المضاءة إلى الحافظة:

```
    Clipboard.SetText TxtUserArea.SelText
```

نص الإجراء munCut_Click()

ينفذ الإجراء munCut_Click() عند اختيار المستخدم البند **قص** من قائمة **تعديل:**

```
Private Sub mnuCut_Click()
    Clipboard.Clear
    Clipboard.SetText txtUserArea.SelText
    txtUserArea.SelText = ""
End Sub
```

يقصد بعملية القص، اقتطاع الجزء المضاء من النص ونقله إلى الحافظة:

تمحو أول عبارة في الإجراء محتويات الحافظة:

```
    Clipboard.Clear
```

ثم تنسخ العبارة الثانية النص المنتقى إلى الحافظة:

```
    Clipboard.SetText TxtUserArea.SelText
```

ثم تقتطع العبارة الثالثة النص المضاء:

```
TxtUserArea.SelText = ""
```

نص الإجراء munPaste_Click()

ينفذ هذا الإجراء عند اختيار البند **لصق** من قائمة **تعديل**:

```
Private Sub mnuPaste_Click()
    txtUserArea.SelText = Clipboard.GetText()
End Sub
```

يقصد بعملية اللصق استبدال المنطقة المضاءة من مربع النص بمحتويات الحافظة. أو حشر محتويات الحافظة في مربع النص مكان وجود المشيرة إذا لم يكن المستخدم قد اختار جزءً من النص. وينجز ذلك بواسطة العبارة التالية:

```
TxtUserArea.SelText = Clipboard.GetText()
```

الخاصية SelLength

تعرّف هذه الخاصية كمتحول من النوع Long، وتحتوي على عدد الأحرف المضاءة حالياً (أي المنتقاة، والانتقاء أو الاختيار يتم بالضغط على المفتاح Shift من لوحة المفاتيح وتحريك الفأرة إلى نهاية الجزء المطلوب اختياره ثم تحرير المفتاح Shift).

ورغم أن برنامج الحافظة لا يستخدم هذه الخاصية إلا أنك قد تجد لها استخداماً في مشاريع فيجول بيسك المستقبلية.

فمثلاً لتحديد عدد الرموز التي تم اختيارها للتو في مربع النص المدعو txtMyTextBox استخدم العبارة التالية:

```
NumberOfCharacters = txtMyTextBox.SelLength
```

نقل الصور من وإلى الحافظة: برنامج الحافظة2

وضح برنامج الحافظة كيفية نقل نص بين الحافظة ومربع النص. تمتاز الحافظة بقدرتها على احتواء الصور أيضاً. يوضح برنامج الحافظة2 كيفية ذلك:

التمثيل المرئي لبرنامج الحافظة2

سنبدأ كعادتنا بطور التمثيل المرئي لنموذج البرنامج:

- أنشئ مشروعاً من النوع Standard EXE.

- احفظ نموذج المشروع باسم AnyData.Frm في الدليل C:\VB5Prg\Ch11 واحفظ ملف المشروع باسم AnyData.Vbp في ذات الدليل.
- أنشئ النموذج frmAnyData طبقاً للجدولين 11-3 و 11-4.

يفترض أن يبدو النموذج المكتمل كذاك المبين في الشكل 11-2.



الشكل 11-2

النموذج frmAnyData.

الجدول 11-3. جدول خصائص برنامج الحافظة2.

الكائن	الخاصية	القيمة
Form	**Name**	**frmAnyData**
	Caption	برنامج الحافظة 2
	RightToLeft	True
PictureBox	**Name**	**picMyPicture**
ListBox	**Name**	**lstList**
	RightToLeft	True
ComboBox	**Name**	**cboList**
	Text	(اتركه فارغاً)
	RightToLeft	True
TextBox	**Name**	**txtUserArea**

	Text	(اتركه فارغاً)
	MultiLine	True
	ScrollBars	3-Both
	RightToLeft	True
Menu	(انظر الجدول 11-4)	(انظر الجدول 11-4)

الجدول 11-4. جدول قائمة برنامج الحافظة2.

مفتاح الاختزال Shortcut	الاسم	العنوان
None	mnuFile	&ملف
None	mnuExit	&خروج...
None	mnuEdit	&تعديل
Ctrl+C	mnuCopy	&نسخ...
Ctrl+X	mnuCut	&قص...
Ctrl+V	mnuPaste	&لصق...

إدخال نص برنامج الحافظة2

سندخل الآن نص برنامج الحافظة2:

- كالعادة يجب أن يحوي قسم التصاريح العامة على العبارة التالية:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء ()Form_Load:

```
Private Sub Form_Load()
    cboList.AddItem "كمبيوتر"
    cboList.AddItem "شاشة"
    cboList.AddItem "طابعة"
    lstList.AddItem "واحد"
    lstList.AddItem "اثنان"
    lstList.AddItem "ثلاثة"
End Sub
```

- أدخل النص التالي ضمن الإجراء picMyPicture_GotFocus():

```
Private Sub picMyPicture_GotFocus()
    picMyPicture.BorderStyle = 1
End Sub
```

- أدخل النص التالي ضمن الإجراء picMyPicture_LostFocus():

```
Private Sub picMyPicture_LostFocus()
    picMyPicture.BorderStyle = 0
End Sub
```

- أدخل النص التالي ضمن الإجراء mnuCopy_Click():

```
Private Sub mnuCopy_Click()
    Clipboard.Clear
    If TypeOf Screen.ActiveControl Is TextBox Then
      Clipboard.SetText Screen.ActiveControl.SelText
    ElseIf TypeOf Screen.ActiveControl Is ComboBox Then
      Clipboard.SetText Screen.ActiveControl.Text
    ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
      Clipboard.SetText Screen.ActiveControl.Picture
    ElseIf TypeOf Screen.ActiveControl Is ListBox Then
      Clipboard.SetText Screen.ActiveControl.Text
    Else
    End If
End Sub
```

- أدخل النص التالي ضمن الإجراء mnuCut_Click():

```
Private Sub mnuCut_Click()
    mnuCopy_Click
    If TypeOf Screen.ActiveControl Is TextBox Then
      Screen.ActiveControl.SelText = ""
    ElseIf TypeOf Screen.ActiveControl Is ComboBox Then
      Screen.ActiveControl.SelText = ""
    ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
      Screen.ActiveControl.Picture = LoadPicture()
    ElseIf TypeOf Screen.ActiveControl Is ListBox Then
      If Screen.ActiveControl.ListIndex >= 0 Then
        Screen.ActiveControl.RemoveItem _
            Screen.ActiveControl.ListIndex
      End If
```

```
    Else
    End If
End Sub
```

- أدخل النص التالي ضمن الإجراء ()mnuPaste_Click:

```
Private Sub mnuPaste_Click()
    If TypeOf Screen.ActiveControl Is TextBox Then
      Screen.ActiveControl.SelText = Clipboard.GetText()
    ElseIf TypeOf Screen.ActiveControl Is ComboBox Then
      Screen.ActiveControl.Text = Clipboard.GetText()
    ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
      Screen.ActiveControl.Picture = Clipboard.GetData()
    ElseIf TypeOf Screen.ActiveControl Is ListBox Then
      Screen.ActiveControl.AddItem Clipboard.GetText()
    Else
    End If
End Sub
```

- أدخل النص التالي ضمن الإجراء ()munExit_Click:

```
Private Sub mnuExit_Click()
    End
End Sub
```

- احفظ المشروع باختيار البند **Save Project** من قائمة **File**.

تنفيذ برنامج الحافظة2

لنشاهد ما كتبناه قيد التنفيذ:

- نفّذ برنامج الحافظة2.

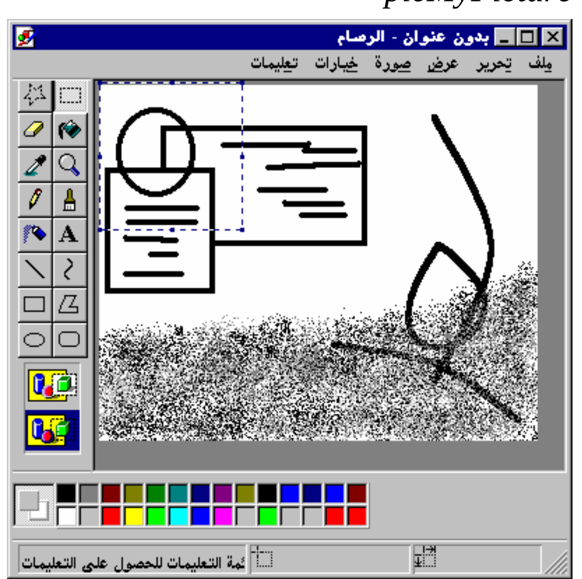
يمكّنك البرنامج من نسخ وقص ولصق المعطيات من وإلى الحافظة. والمعطيات قد تكون صوراً أو نصوصاً.

اتبع الخطوات التالية لنسخ صورة من برنامج الرسام Paint إلى مربع الصورة الموجود في برنامج الحافظة2:

- نفّذ برنامج الرسام أثناء عمل برنامج الحافظة2.
- ارسم شيئاً ما في برنامج الرسام.

- انسخ جزءاً من الشكل المرسوم إلى الحافظة، بتحديد ذلك الجزء، ثم اختيار البند **نسخ** من قائمة **عرض** التابعة لبرنامج الرسام. يوضح الشكل 11-3 جزء من الصورة تم نسخه إلى الحافظة.
- انتقل إلى برنامج الحافظة2.
- تحقق بأن عنصر التحكم picMyPicture فعّال الآن (انقر عليه بواسطة الفأرة).
- اختر البند **لصق** من قائمة **تعديل** التابعة لبرنامج الحافظة2.

يستجيب برنامج الحافظة2 بنسخ الصورة المقيمة في الحافظة إلى مربع الصورة picMyPicture.



الشكل 11-3
استخدام برنامج الرسام لرسم صورة بهدف نسخها إلى برنامج الحافظة2.



الشكل 11-4
نسخ الصورة إلى برنامج الحافظة2.

- يمكّنك برنامج الحافظة2 أيضاً من نسخ وقص ولصق النصوص من وإلى مربع

النص أو مربع السرد أو مربع السرد والتحرير.

- تمرّن على فعاليات برنامج الحافظة2. لا تستطيع لصق الصورة في مربع السرد أو مربع النص أو في مربع السرد والتحرير. كذلك لا تستطيع لصق نص في مربع الصورة. وقبل أن تتمكن من لصق المعطيات في كائن ما، لا بد من اختيار ذلك الكائن أولاً بالنقر عليه (يجب أن ينتقل التركيز إليه).

كيف يعمل برنامج الحافظة 2

يستخدم برنامج الحافظة2 الإجراءات munCopy_Click() و munCut_Click() و munPaste_Click() لنقل المعطيات من وإلى الحافظة Clipboard.

نص الإجراء Form_Laod()

ينفذ الإجراء Form_Laod() آلياً عند تشغيل البرنامج:

```
Private Sub Form_Load()
    cboList.AddItem "كمبيوتر"
    cboList.AddItem "شاشة"
    cboList.AddItem "طابعة"

    lstList.AddItem "واحد"
    lstList.AddItem "اثنان"
    lstList.AddItem "ثلاثة"
End Sub
```

يملأ هذا الإجراء كل من مربع السرد ومربع السرد والتحرير بثلاثة بنود.

نص الإجراء picMyPicture_GotFocus()

ينفذ الإجراء picMyPicture_GotFocus() عندما ينتقل التركيز إلى مربع الصورة (أي عند اختيار مربع الصورة):

```
Private Sub picMyPicture_GotFocus()

    picMyPicture.BorderStyle = 1

```

```
End Sub
```

ما الفائدة من هذا الإجراء ؟ !

لا يعطي فيجول بيسك أي دلالة واضحة عند اختيار مربع الصورة، بخلاف ما هو الحال عليه في مـــربع النص أو مربع السرد أو مربع السرد والتحرير. ولهذا فإن العبارة التي أدخلناها ضمن الإجراء picMyPicture_GotFocus()، تبدل الخاصية BorderStyle بحيث تتمكن من تمييز أن مربع الصورة تم انتقاؤه. إذ أن إسناد القيمة 1 إلى الخاصية BorderStyle يتسبب بوضع حافة لعنصر تحكم الصورة.

نص الإجراء picMyPicture_LostFocus()

ينفذ هذا الإجراء عند عدم اختيار مربع الصورة (بعكس الإجراء السابق تماماً):

```
Private Sub picMyPicture_LostFocus()
    picMyPicture.BorderStyle = 0
End Sub
```

إذاً، أسندت العبارة الوحيدة في هذا الإجراء القيمة صفر إلى الخاصية BorderStyle، ويتسبب ذلك بإزالة الحافة المحيطة بعنصر تحكم الصورة، وهذا ما يساعد على تمييز أن مربع الصورة ليس منتقىً الآن.

نص الإجراء munCopy_Click()

يُنفذ هذا الإجراء عندما يختار المستخدم البند **نسخ** من قائمة **تعديل**. يمحو نص هذا الإجراء محتويات الحافظة وينسخ المحتويات المضاءة إلى الحافظة:

```
Private Sub mnuCopy_Click()
    Clipboard.Clear
    If TypeOf Screen.ActiveControl Is TextBox Then
      Clipboard.SetText Screen.ActiveControl.SelText
    ElseIf TypeOf Screen.ActiveControl Is ComboBox Then
      Clipboard.SetText Screen.ActiveControl.Text
    ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
      Clipboard.SetText Screen.ActiveControl.Picture
    ElseIf TypeOf Screen.ActiveControl Is ListBox Then
      Clipboard.SetText Screen.ActiveControl.Text
```

```
    Else
    End If
End Sub
```

إذاً يمحو هذا الإجراء محتويات الحافظة أولاً، ثم ينسخ محتويات عنصر التحكم الفعّال إلى الحافظة، ولهذا يتوجب أولاً معرفة نوع العنصر الفعال. ينجز الإجراء ذلك باستخدام سلسلة من عبارات الشرط If TypeOf.

تتحقق أول عبارة شرطية If TypeOf إذا كان عنصر التحكم الفعّال هو مربع نص، ويتم نسخ النص المضاء في مربع النص إلى الحافظة:

```
Clipboard.SetText Screen.ActiveControl.SelText
```

أما إذا كان العنصر الفعال هو مربع السرد والتحرير، فيتم نسخ النص المضاء إلى الحافظة:

```
Clipboard.SetText Screen.ActiveControl.Text
```

وإذا كان العنصر الفعال هو مربع صورة، يتم نسخ الصورة إلى الحافظة.

```
Clipboard.SetData Screen.ActiveControl.Picture
```

وأخيراً إذا كان العنصر الفعال هو عنصر السرد، يتم نسخ النص المضاء إلى الحافظة:

```
Clipboard.SetText Screen.ActiveControl.Text
```

ملاحظة

تمثل قيمة المتحول Screen.ActiveControl عنصر التحكم الفعّال حالياً. يجدّد البرنامج آلياً هذا المتحول أثناء التنفيذ، وكما ترى يعتبر هذا المتحول مفيداً لإنجاز المهام على عنصر التحكم الفعّال الحالي.

نص الإجراء munCut_Click()

يُنفذ هذا الإجراء عند اختيار البند قص من قائمة تعديل:

```
Private Sub mnuCut_Click()
    mnuCopy_Click
    If TypeOf Screen.ActiveControl Is TextBox Then
      Screen.ActiveControl.SelText = ""
    ElseIf TypeOf Screen.ActiveControl Is ComboBox Then
```

```
  Screen.ActiveControl.SelText = ""
ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
  Screen.ActiveControl.Picture = LoadPicture()
ElseIf TypeOf Screen.ActiveControl Is ListBox Then
  If Screen.ActiveControl.ListIndex >= 0 Then
    Screen.ActiveControl.RemoveItem _
        Screen.ActiveControl.ListIndex
  End If
Else
End If
End Sub
```

يعرّف القص، بأنه نسخ أولاً للمعطيات المنتقاة، ثم حذفها من الأصل. ولهذا ينفذ هذا الإجراء أولاً، الإجراء munCopy_Click():

```
mnuCopy_Click
```

ثم يستخدم سلسلة من عبارات If TypeOf لمعرفة نوع عنصر التحكم الفعال الحالي وتقرير طريقة حذف المعطيات التي تم نسخها.

فإذا كان عنصر التحكم الفعال الحالي هو مربع نص، يحُذف النص المنسوخ بواسطة العبارة:

```
Screen.ActiveControl.SelText = ""
```

والشكل مشابه بالنسبة لمربع السرد والتحرير، حيث يحُذف النص المنسوخ بواسطة العبارة:

```
Screen.ActiveControl.Text = ""
```

بينما يُستخدم التابع الوظيفي LoadPicture() لمحو مربع الصورة.

```
Screen.ActiveControl.Picture = LoadPicture()
```

وبسبب عدم تحديد اسم صورة لتحميلها بواسطة التابع الوظيفي LoadPicture() (وذلك بعدم ذكر اسم الصورة بين القوسين وتركهما فارغين). يمحو هذا التابع الصورة الحالية، وهو بالضبط ما نبتغيه.

أما إذا كان عنصر التحكم الفعال هو مربع سرد. فلا بد قبل حذف العنصر، من التأكد أن قيمة الخاصية ListIndex أكبر من أو تساوي الصفر. تمثل الخاصية ListIndex رقم البند المنتقى حالياً. ولهذا يجب التحقق من وجود عنصر منتقى حالياً في مربع السرد

(فمثلاً إذا كانت قيمة الخاصية ListIndex تساوي −1 فهذا يعني أنه لا يوجد عنصر منتقى حالياً في مربع السرد).

نص الإجراء ()munPaste_Click

ينفذ هذا الإجراء عند اختيار البند **لصق** من قائمة **تعديل**:

```
Private Sub mnuPaste_Click()
    If TypeOf Screen.ActiveControl Is TextBox Then
      Screen.ActiveControl.SelText = Clipboard.GetText()
    ElseIf TypeOf Screen.ActiveControl Is ComboBox Then
      Screen.ActiveControl.Text = Clipboard.GetText()
    ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
      Screen.ActiveControl.Picture = Clipboard.GetData()
    ElseIf TypeOf Screen.ActiveControl Is ListBox Then
      Screen.ActiveControl.AddItem Clipboard.GetText()
    Else
    End If
End Sub
```

يستخدم هذا الإجراء كسابقيه، سلسلة من عبارات If TypeOf لتحديد نوع العنصر الفعّال الحالي، ويستخدم الطريقة المناسبة لنقل المعطيات إلى العنصر الفعال الحالي.

استخدام التابع الوظيفي ()GetFormat لتحديد نوع المعطيات الموجودة في الحافظة

كما ذكرنا، تعتبر الحافظة قادرة على الإمساك بالصور والنصوص على حد سواء. يستخدم التابع الوظيفي ()GetFormat لمعرفة نوع المعطيات الموجودة حالياً في الحافظة.

فمثلاً، تحدد العبارة التالية، ما إذا كانت المعطيات الموجودة في الحافظة نصية أم لا:

```
If Clipboard.Getformat (vbCFText) Then
   'يوجد في الحافظة معلومات نصية
End If
```

يعيد التابع (GetFormat (vbCFText القيمة True إذا كانت الحافظة تحوي نصاً.

أما لتحديد ما إذا كانت الحافظة تحوي صورة نقطية، فاستخدم العبارة التالية:

```
If Clipboard.Getformat (vbCFBitmap) Then
```

```
'يـوجـد فـي الحـافـظـة صـورة نـقـطـيـة
End If
```

يعيد (GetFormat (vbCFBitmap القيمة True إذا كانت الحافظة ممسكة بصورة.

الزمن الضائع Idle Time

تمر لحظات كثيرة أثناء تنفيذ برامج فيجول بيسك، بحيث لا ينفذ فيها أي شيء (وقت ضائع). خذ مثالاً، برنامج يصدر صوت رنين Beep مائة مرة، ثم ينتظر إعادة نقر زر يدعى **رنين** لإصدار صوت الرنين مجدداً. يمر البرنامج في زمن ضائع بانتظار نقر أو ضغط الزر **رنين** وقد يطول هذا الوقت الضائع.

تستطيع إنجاز مهام أخرى أثناء مرور الوقت الضائع، بالتحول إلى برامج أخرى في ويندوز، فمثلاً عندما يكون البرنامج السابق متوقفاً عن إصدار الرنين، تستطيع حينها الانتقال إلى برامج أخرى، ومتابعة العمل.

أما عندما تنقر الزر **رنين**، ليبدأ البرنامج بإصدار الرنين مائة مرة، حينها لا تستطيع الانتقال إلى برامج أخرى في ويندوز، إلى أن ينتهي البرنامج من إصدار الرنين لمائة مرة.

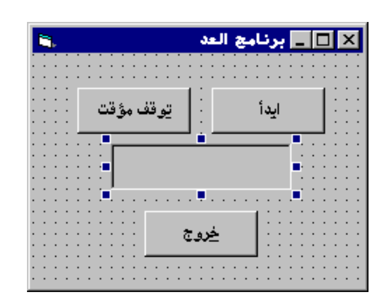
برنامج العد

سنكتب الآن برنامجاً يدعى برنامج العد، لعد الأرقام من 1 إلى 999.

التمثيل المرئي لبرنامج العد

- أنشئ مشروعاً جديداً بنوع StandardEXE.
- احفظ نموذج المشروع باسم Count.Frm في الدليل C:\VB5Prg\Ch11 واحفظ ملف المشروع باسم Count.Vbp في ذات الدليل.
- أنشئ نموذج برنامج العد تبعاً للجدول 11-5.

يُفترض أن يبدو النموذج كما في الشكل 11-5.



الشكل 11-5

النموذج frmCount.

الجدول 11-5. جدول خصائص برنامج العد.

الكائن	الخاصية	القيمة
Form	**Name**	**frmCount**
	Caption	برنامج العد
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdPause**
	Caption	&توقف مؤقت
	RightToLeft	True
CommandButton	**Name**	**cmdStart**
	Caption	ا&بدأ
	RightToLeft	True
Label	**Name**	**lblResult**
	Alignment	2-Center
	BorderStyle	1-Fixed Single
	Caption	(اجعله فارغاً)
	RightToLeft	True

إدخال نص برنامج العد

- تحقق من وجود العبارة Option Explicit ضمن قسم التصاريح العامة.

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء ()cmdExit_Click:

```
Private Sub cmdExit_Click()
    End
End Sub
```

- أدخل النص التالي ضمن الإجراء ()cmdStart_Click:

```
Private Sub cmdStart_Click()
    Dim Counter As Integer
    Counter = 1
    ' Counter from 1 to 999
    Do While Counter < 1000
      1b1Result.Caption = Str$(Counter)
      Counter = Counter + 1
    Loop
End Sub
```

ملاحظة

لاحظ أننا لم نطلب منك حتى هذه النقطة، كتابة أي شيء ضمن الإجراء ()cmdPause_Click.

تنفيذ برنامج العد

لنشاهد أثر ما كتبناه:

- نفّذ برنامج العد.
- انقر الزر **ابدأ العد**.

يبدو البرنامج وكأنه لا يفعل شيئاً لفترة، ثم يظهر الرقم 999. ويعتمد سرعة ذلك كله على سرعة الحاسب المستخدم.

قد تتوقع لدى قراءتك لنص الإجراء ()cmdStart_Click أن البرنامج سيظهر كل الأرقام من 1 إلى 999، ولفهم السبب الذي منع البرنامج من ذلك، تأمل العبارات المسئولة عن إظهار الأرقام. هذه هي العبارة المسئولة عن تغيير عنوان اللافتة:

```
lblResult.Caption = Strs(Counter)
```

لفيجول بيسك قدرة على إنعاش الشاشة (وهي معدل إعادة تجديد محتويات الشاشة Refresh Rate) فقط أثناء الزمن الضائع (أي عندما ينتهي تنفيذ الإجراء ويعود البرنامج إلى الزمن الضائع Idle Time).

تذكر أن نقر زر **خروج** أثناء التعداد لا يُنهي البرنامج، لأن البرنامج يستطيع تمييز حوادث النقر فقط أثناء الأزمنة الضائعة. وحالما ينتهي التعداد يعود البرنامج إلى الزمن الضائع ويستجيب لحادثة نقر الزر **خروج**.

ملاحظة

إذا كان الحاسب الذي تستخدمه سريعاً، فلعل استبدال العبارة:

Do While Counter < 1000

في الإجراء ()cmdStart_Click بالعبارة:

Do While Counter < 10000

يساعدك على تمييز الحلقة بشكل أفضل، لأنك قد لا تتمكن من مشاهدة تنفيذ الحلقة مع القيم الصغيرة إذا كان حاسبك بالغ السرعة (Pentium MMX).

تعديل برنامج العد

اتبع الخطوات التالية لتعديل برنامج العد بحيث تُظهر اللافتة كل رقم من 1 وحتى 999 أثناء التعداد:

- أضف العبارة التالية إلى الإجراء ()cmdStart_Click:

```
lblResult.Refresh
```

يفترض أن يبدو الإجراء ()cmdStart_Click كما يلي:

```
Private Sub cmdStart_Click()
    Dim Counter As Integer
    Counter = 1
    ' Counter from 1 to 999
    Do While Counter < 1000
      1b1Result.Caption = Str$(Counter)
       lblResult.Refresh
      Counter = Counter + 1
    Loop
End Sub
```

تتسبب الطريقة Refresh بإعادة إنعاش الشاشة فوراً.

استخدم الطريقة Refresh للتسبب بإعادة إنعاش أي كائن:

```
Object.Refresh
```

تنفيذ نسخة برنامج العد المحسنة

لنشاهد تأثير ما كتبناه:

- نفّذ نسخة برنامج العد المحسنة.

*تُظهر اللافتة الآن كل رقم من 1 وحتى 999 أثناء التعداد. لاحظ أن النقر على **خروج** أثناء التعداد لا يُنهي البرنامج، ما هو السبب ؟ ! السبب أن البرنامج أثناء تنفيذ الإجراء cmdStart_Click() ليس في زمن ضائع (ولهذا فالبرنامج لا يمكنه الاستجابة لنقر الزر **خروج**).*

المزيد من التحسين على برنامج العد

لنحسن برنامج العد بحيث يستجيب للنقر بواسطة الفأرة على الزر **خروج** أثناء التعداد. سنحتاج إلى كتابة إجراء يدعى Main()، والذي سيكون أول إجراء يُنفذ عند تشغيل البرنامج.

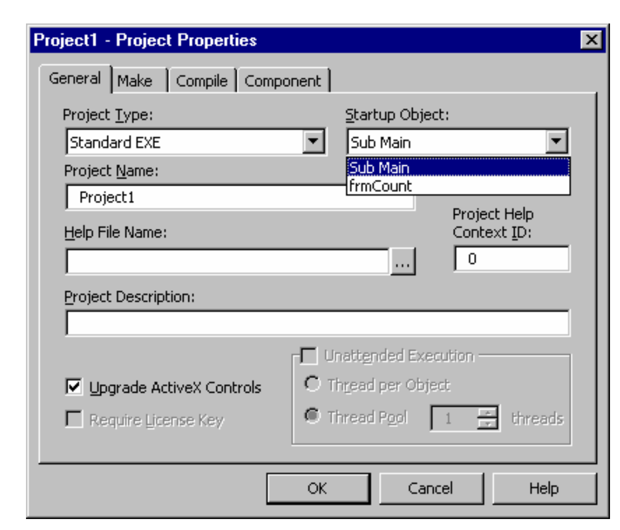
- اختر البند **Project1 Properties** من قائمة **Project**.

يستجيب فيجول بيسك بإظهار نافذة خصائص المشروع.

- اختر الصفحة General في مربع الحوار السابق.

ألقِ نظرة على الحقل المدعو Startup Object، والذي يظهر في صفحة General، وبما أنك ترغب أن يكون الإجراء Main() هو أول إجراء ينفذ عند تشغيل برنامج العد، لهذا غيّر الحقل Startup Object إلى Main() وفق ما يلي:

- ضع الحقل Startup Object على Sub Main، ثم انقر الزر **Ok** (انظر الشكل 6-11).



الشكل 11-6

وضع الحقل Startup Object

على Sub Main.

والآن، الإجراء Main() هو أول إجراء ينفذ عند تشغيل برنامج العد.

سنكتب الآن الإجراء Main(). يحتاج فيجول بيسك إلى أن يكون Main() ضمن وحدة نمطية (Module) مستقلة. لهذا السبب أضف وحدة نمطية جديدة إلى المشروع:

- اختر البند **Add Module** من قائمة **Project**.

يستجيب فيجول بيسك بإظهار مربع الحوار Add Module.

- اختر الرمز Module في صفحة New لمربع الحوار Add Module ثم انقر الزر **فتح**.

*يستجيب فيجول بيسك بإضافة وحدة نمطية Module جديدة إلى المشروع. الاسم الافتراضي للوحدة النمطية المضافة حديثاً هو Module1.Bas. يمكنك مشاهدة ذلك في إطار المشروع باختيار البند **Project Explores** من قائمة **View**.*

- احفظ الوحدة النمطية المضافة حديثاً باسم CountM.Bas كما يلي:
- تحقق من توضع الإضاءة على Module1.Bas في إطار المشروع ثم اختر **Save Module1 As** من قائمة **File** لفيجول بيسك.

يستجيب فيجول بيسك بإظهار مربع الحوار Save File As.

- احفظ الوحدة النمطية المضافة حديثاً باسم CountM.Bas في الدليل C:\VB5Prg\Ch11.

سيحتوي إطار المشروع الآن على Count.Frm و CountM.Bas.

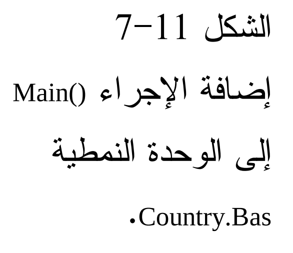
سنضيف إجراءً يدعى Main() في الجزء المدعو CountM.Bas:

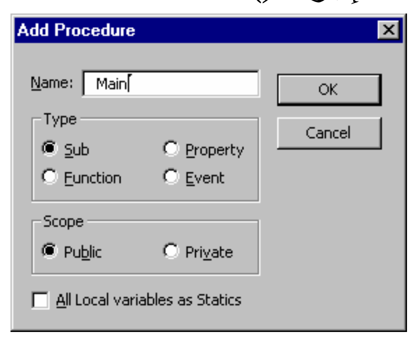
- تأكد من تمركز الإضاءة على CountM.Bas في الإطار Project.
- انقر على الرمز View Code الذي يظهر أقصى يسار الرموز على شريط الأدوات
  Tool Bar لإطار المشروع (لأنك على وشك إضافة إجراء إلى الوحدة النمطية CountM.Bas).
- اختر البند **Add Procedure** من قائمة **Tools**.

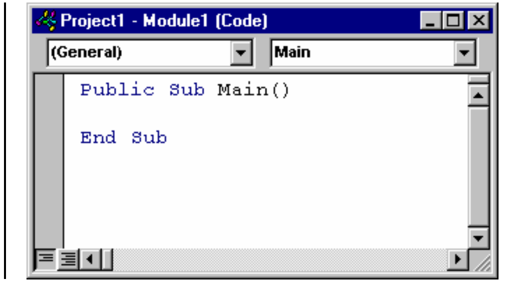
يستجيب فيجول بيسك بإظهار مربع الحوار Add Procedure.

- اكتب Main في الحقل Name لمربع الحوار Add Procedure، وتحقق بأن Type موضوع على Sub وأن Scope موضوع على Public (انظر الشكل 11-7)، وأخيراً انقر الزر **Ok**.

يستجيب فيجول بيسك بإضافة الإجراء Main() في CountM.Bas. يبين الشكل 11-8 الإجراء Main().


الشكل 11-7
إضافة الإجراء Main()
إلى الوحدة النمطية
Country.Bas.





الشكل 11-8
الإجراء Main().

- أدخل النص التالي في قسم التصاريح العامة:

```
'يجب التصريح عن كل المتحولات
```

```
Option Explicit
```

- أدخل النص التالي في الإجراء Main() الموجود في الوحدة النمطية CountM.Bas:

```
Public Sub Main()
    Dim Counter
    frmCount.Show
    Do While DoEvents()
      If ggflag = 1 Then
      Counter = Counter + 1
      frmCount.lblResult.Caption = Str$(Counter)
         If Counter = 999 Then
          Counter = 1
         End If
      End If
    Loop
End Sub
```

- أدخل النص التالي في الإجراء cmdPause_Click() للنموذج frmCount:

```
Private Sub cmdPause_Click()
    ggflag = 0
End Sub
```

- بدل نص الإجراء cmdStartr_Click() للنموذج frmCount بحيث يغدو كما يلي:

```
Private Sub cmdStart_Click()
    ggflag = 1
End Sub
```

- أدخل النص التالي ضمن الإجراء Form_Load() للنموذج frmCount:

```
Private Sub Form_Load()
    ggflag = 0
End Sub
```

- احفظ المشروع باختيار البند **Save Project** من قائمة **File** لفيجول بيسك.

تنفيذ النسخة المحسنة من برنامج العد

لنشاهد أثر ما كتبناه:

- نفّذ نسخة برنامج العد المحسنة.

يعد البرنامج من 1 إلى 999 ثم يكرر التعداد ... وهكذا. ويظهر كل رقم أثناء التعداد في اللافتة.

- انقر الزر **إيقاف مؤقت**.

كما تلاحظ، تسبب ذلك بإيقاف التعداد.

- انقر الزر **ابدأ** لاستئناف التعداد.

افتح برامج أخرى أثناء قيام برنامج العد بالتعداد وأنجز ما ترغب من الأعمال، كأن تفتح نافذة الرسام وترسم شكلاً ما (انظر الشكل 11-9) لاحظ أن البرنامج يعد أثناء عملك ضمن برنامج الرسام.



الشكل 11-9
تنفيذ برنامج العد
وبرنامج الرسام سوية.

الإجراء Main()

الإجراء Main() هو أول إجراء يُنفذ عند تشغيل البرنامج.

ولهذا فإن العبارة الأولى فيه مسئولة عن إظهار النموذج:

```
frmCount.Show
```

ملاحظة

استخدم الطريقة Show كأول عبارة في الإجراء Main() لإظهار النموذج. لأن

الإجراء ()Main هو أول إجراء يُنفذ عند تشغيل البرنامج، ولن يحمل النموذج آلياً.

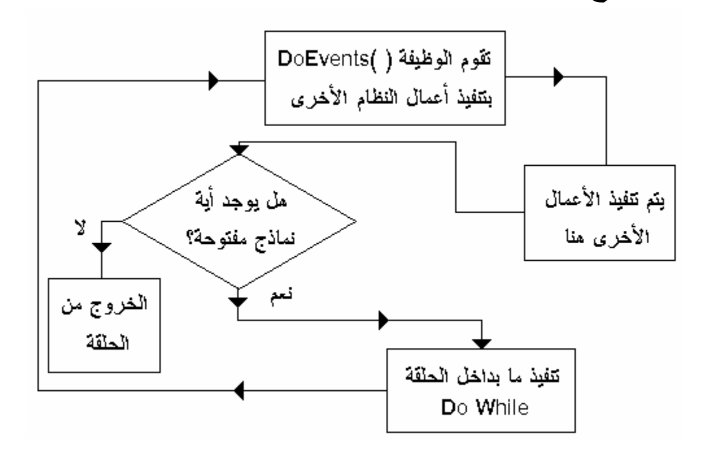
تدعى الحلقة ()Do While DoEvents بحلقة الوقت الضائع idle Loop:

```
Public Sub Main()
    Dim Counter
    frmCount.Show
    Do While DoEvents()
      If ggflag = 1 Then
      Counter = Counter + 1
      frmCount.lblResult.Caption = Str$(Counter)
         If Counter = 999 Then
          Counter = 1
         End If
      End If
    Loop
End Sub
```

يوضح الشكل 11–10 الانسياب المنطقي لهذه الحلقة. لاحظ أن البرنامج يبقى في حالة دوران لا نهائية.

إنهاء الحلقة الغير المنتهية

يعيد التابع الوظيفي ()DoEvents عدد النماذج المفتوحة حالياً. فإذا أغلقت النموذج frmCount، فهذا يعني عدم تبقي نماذج مفتوحة في البرنامج، وبالتالي يعيد التابع الوظيفي ()DoEvents القيمة صفر. يتسبب هذا بإنهاء الحلقة، والتي تنهي بدورها البرنامج.



الشكل 11–10
حلقة الوقت الضائع
.Do While DoEvents()

تنفيذ جسم الحلقة ()Do While DoEvents

حسب ما يوضحه الشكل 11–10 ينفذ جسم هذه الحلقة في حال وجود نموذج مفتوح. العبارة الأولى في الحلقة، هي عبارة If التي تتحقق من قيمة المتحول ggFlag. لنفترض أنك نقرت على الزر **ابدأ** للتو، مما يؤدي إلى إسناد القيمة 1 للمتحول ggFlag، وبالتالي تُنفذ عبارتي If وتزيد قيمة المتحول، وتجدّد خاصية Caption للافتة.

لا يمر البرنامج في زمن ضائع أثناء تنفيذ جسم الحلقة Do While. مما يعني أن البرنامج لا يستطيع الاستجابة إلى حوادث معينة أثناء تنفيذ الحلقة هذه، فمثلاً. لا يستجيب لحادثة الضغط على الزر **إيقاف مؤقت**. تدعى الحوادث التي تجري في غير الأزمنة الضائعة بالحوادث المعلقة Pending Events. فمثلاً النقر على الزر **إيقاف مؤقت** أثناء تنفيذ البرنامج لجسم الحلقة Do While يسبب حادثة معلقة.

يحتوي جسم الحلقة Do While على العبارة التالية:

```
frmCount.lblResult.Caption = Str$(Counter)
```

تتسبب هذه العبارة بظهور حادثة معلقة، لأن الخاصية Caption للافتة lblResult، لا يمكن تعديلها إلا في الزمن الضائع فقط.

حسب ما يوضحه الشكل 11–10، حالما تُنفذ الحلقة Do While، يعيد البرنامج تنفيذ الحلقة ()DoEvents مرة ثانية (إلى ما لا نهاية).

التابع الوظيفي ()DoEvents

يتوجب عليك إجبار البرنامج على الدخول في زمن ضائع، حتى يتمكن النظام ويندوز من التعامل مع الحوادث المعلقة، وهذا هو دور التابع الوظيفي ()DoEvents.

يتعامل البرنامج مع الحوادث المعلقة عندما يمر في زمن ضائع. فمثلاً عملية إعادة إنعاش اللافتة IblResult حادثة معلقة، كذلك النقر على الزر **إيقاف مؤقت** أثناء تنفيذ الحلقة Do While عملية معلقة أيضاً. ولهذا فإن استخدام ()DoEvents يسمح للبرنامج بالدخول في زمن ضائع، مما يسمح له بمعالجة الحوادث المعلقة. ثم يعود البرنامج إلى تنفيذ الحلقة Do While وتبدأ المعالجة كاملة من جديد.

ملاحظة

يقوم التابع DoEvents() عند استخدامه مع الحلقة Do While بوظيفتين:- القيمة المعادة من هذا التابع تمثل عدد النماذج المفتوحة حالياً. وتنهى الحلقة عند عدم وجود أي نماذج مفتوحة.- يُجبر هذا التابع الوظيفي البرنامج، على الدخول في زمن ضائع، مما يسمح لويندوز بتنفيذ الحوادث المعلقة.

نص قسم التصاريح العامة للوحدة النمطية CountM.Bas

يتضمن هذا القسم العبارتين:

التصريح عن المتحول ggFlag في هذا المكان يعني أنه مرئي (متاح) لكل الإجراءات الموجودة في الوحدات النمطية الأخرى أو النماذج. ويستخدم هذا المتحول كل من الإجراءات Form_Load() و cmdStart_Click() و cmdPause_Click() للنموذج frmCount كما يستخدم من قبل الإجراء Main() الموجود في الوحدة النمطية CountM.Bas.

نص الإجراء cmdPause_Click()

يُنفذ هذا الإجراء أثناء الزمن الضائع، إذا نقرت الزر **إيقاف مؤقت** خلال تنفيذ الحلقة Do While DoEvents(). يسند هذا الإجراء للمتحول ggFlag القيمة صفر، مما يتسبب بإيقاف عملية التعداد في الإجراء Main:

```
Private Sub cmdPause_Click()
    ggflag = 0
End Sub
```

نص الإجراء cmdStart_Click()

يسند هذا الإجراء للمتحول ggFlag القيمة واحد، مما يتسبب بعمل معالجة التعداد في الإجراء Main():

```
Private Sub cmdStart_Click()
    ggflag = 1
End Sub
```

الخلاصة

تعلمنا فى هذا الفصل كيف يتمكن برنامج مكتوب بفيجول بيسك من الإستفادة من مظاهر ويندوز الأساسية. فتعلمنا كيفية إنجاز نسخ وقص ولصق (صور أو نصوص) من وإلى الحافظة، كما تعلمنا استخدام الطريقة Refresh اللازمة لإعادة رسم الكائنات، وكيفية كتابة نص برنامج يمكن تنفيذه فى الأزمنة الضائعة.

الفصل الثاني عشر

لوحة المفاتيح

سنتعلم في هذا الفصل كيف يستجيب برنامجك لحوادث لوحة المفاتيح، وكيفية التحقق من الضغط على أحد المفاتيح أو تحريره، وآلية معالجة معطيات الإدخال الواردة من لوحة المفاتيح.

تركيز لوحة المفاتيح Focus

كما تبين لنا من الفصول السابقة، الكائن الذي يمتلك تركيز لوحة المفاتيح هو الكائن الذي يستجيب لإدخالها. وعندما يكون تركيز لوحة المفاتيح عند عنصر تحكم ما، فإنه سوف يُبدي تغيراً ما في شكله على الشاشة. فمثلاً عندما يكون التركيز على زر أمر، يظهر مستطيل منقط حول عنوان زر الأمر هذا، وعندما يكون التركيز موضوعاً عند شريط تمرير، فسوف تجد أن مؤشره يومض.

حوادث لوحة المفاتيح

تترافق فعاليات لوحة المفاتيح مع الحوادث التالية:

- KeyDown (ضغط مفتاح).
- KeyUp (تحرير مفتاح).
- KeyPress (ضغط مفتاح آسكي).

الحادثة KeyDown

تحصل هذه الحادثة، عند ضغط أحد المفاتيح على لوحة المفاتيح إلى الأسفل، مثال، الضغط على أي مفتاح على لوحة المفاتيح، عندما يكون التركيز على زر أمر يدعى cmdPushMe، مما ينتج عنه تنفيذ الإجراء cmdPushMe_KeyDown() آلياً.

الحادثة KeyUp

تحصل هذه الحادثة عند تحرير أحد المفاتيح على لوحة المفاتيح. مثال، تحرير مفتاح ما على لوحة المفاتيح عندما يكون التركيز على زر أمر يدعى cmdPushMe، مما ينتج عنه تنفيذ الإجراء cmdPushMe_KeyUp() آلياً.

الحادثة KeyPress

تحصل هذه الحادثة عند الضغط على مفتاح ما، له شفرة آسكي ASCII مرافقة. فمثلاً، إذا ضغطت على المفتاح A عندما يكون التركيز عند زر الأمر المدعو cmdPushMe،

فسوف يُنفذ الإجراء ()cmdPushMe_KeyUp، أما إذا ضغطت على المفتاحF1 مثلاً فإن الحادثة ()cmdPushMe_KeyPress لن تقع، لأن المفتاح F1 لا يملك شفرة ASCII مرافقة.

برنامج المفاتيح

يوضح برنامج المفاتيح كيفية استخدام حوادث لوحة المفاتيح الثلاث الآنفة الذكر.

التمثيل المرئي لبرنامج المفاتيح

نبدأ كعادتنا بطور التمثيل المرئي لنموذج برنامج المفاتيح:

- أنشئ الدليل C:\VB5Prg\Ch12.
- أنشئ مشروعاً جديداً من النوع Standard EXE، واحفظ نموذج المشروع بالاسم Keys.frm في الدليل C:\VB5Prg\Ch12، واحفظ المشروع في ذات الدليل بالاسم Keys.Vbp.
- أنشئ النموذج طبقاً للجدول 12-1.

يُفترض أن يظهر النموذج المكتمل كما في الشكل 12-1.



الشكل 12-1
النموذج frmKeys
في مرحلة التصميم.

الجدول 12-1. جدول خصائص البرنامج frmKeys.

الكائن	الخاصية	القيمة
Form	**Name**	**frmKeys**
	Caption	برنامج المفاتيح
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdPushMe**
	Caption	ا&ضغط هذا الزر

	RightToLeft	True
Label	**Name**	**lblInfo**
	Alignment	2-Center
	BorderStyle	1-Fixed Single
	Caption	(اجعله فارغاً)
	RightToLeft	True

إدخال نص برنامج المفاتيح

- اكتب العبارة Option Explicit في قسم التصاريح العامة للنموذج frmKeys:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- اكتب النص التالي ضمن الإجراء ()cmdExit_Click للنموذج frmKeys:

```
Private Sub cmdExit_Click()
    End
End Sub
```

- أدخل النص التالي ضمن الإجراء ()cmdPushMe_Click:

```
Private Sub cmdPushMe_keydown(KeyCode As Integer,_
                              Shift As Integer)
    lblInfo.Caption = "لقد ضغطت أي مفتاح"
    lblInfo.Caption = lblInfo.Caption + vbCrLf
    lblInfo.Caption = lblInfo.Caption + "KeyCode=" + _
                                     Str(KeyCode)
    lblInfo.Caption = lblInfo.Caption + vbCrLf
    lblInfo.Caption = lblInfo.Caption + "Shift=" + _
                                     Str(Shift)
End Sub
```

- احفظ المشروع باختيار البند **Save Project** من القائمة **File** لفيجول بيسك.

تنفيذ برنامج المفاتيح

لنشاهد أثر ما كتبناه:

- نفّذ برنامج المفاتيح.
- انقر الزر **اضغط هذا الزر**.

يستجيب برنامج المفاتيح بوضع مستطيل منقط حول عنوان الزر، مما يعني أن هذا الزر يمتلك الآن تركيز لوحة المفاتيح (لأننا اخترناه من الخطوة السابقة بالنقر عليه).

- اضغط أي مفتاح على لوحة المفاتيح.

يستجيب البرنامج بإظهار شفرة المفتاح الذي ضغطته، في اللافتة lblInfo، حسب ما يبينه الشكل 12-2.



الشكل 12-2
برنامج المفاتيح والضغط
على أي مفتاح.

- اضغط على مفاتيح أخرى على لوحة المفاتيح، ولاحظ كيف تُظهر اللافتة lblInfo شفرة المفتاح المضغوط.
- اضغط على المفتاح Num Lock، ولاحظ أن البرنامج يعلن بأن القيمة المرافقة لهذا المفتاح تساوي 144، العلاقة بين الأرقام التي تظهر والمفاتيح المضغوطة مشروحة لاحقاً في هذا الفصل.
- اضغط على المفتاح Shift ثم أتبع ذلك بالضغط على المفتاح Ctrl ثم المفتاح Alt، (أي اضغط على المفتاح Alt مع إبقاء الضغط على مفتاح Shift ومفتاح Ctrl).

تُظهر اللافتة الأرقام 1 ثم 3 ثم 7، الصلة بين هذه الأرقام وبين مفاتيح التحكم: Alt و Ctrl و Shift مبينة لاحقاً في هذا الفصل.

- انقر الزر **خروج** لإنهاء برنامج المفاتيح.

كيف يعمل برنامج المفاتيح

يستخدم برنامج المفاتيح الحادثة KeyDown للاستجابة للمفاتيح المضغوطة على لوحة المفاتيح.

نص الإجراء ()cmdPushMe_KeyDown

يُنفذ الإجراء ()cmdPushMe_KeyDown عندما يمتلك الزر **اضغط هذا الزر**، تركيز لوحة المفاتيح، ثم يُضغط أحد المفاتيح من لوحة المفاتيح:

```
Private Sub cmdPushMe_Keydown(KeyCode As Integer,_
                              Shift As Integer)
    lblInfo.Caption = "لقد ضغطت أي مفتاح"
    lblInfo.Caption = lblInfo.Caption + vbCrLf
    lblInfo.Caption = lblInfo.Caption + "KeyCode=" + _
                                         Str(KeyCode)
    lblInfo.Caption = lblInfo.Caption + vbCrLf
    lblInfo.Caption = lblInfo.Caption + "Shift=" + _
                                         Str(Shift)
End Sub
```

يحوي الإجراء ()cmdPushMe_KeyDown وسيطين، يدعى الأول KeyCode، وهو عبارة عن عدد صحيح يمثل شفرة المسح Scan Code للمفتاح المضغوط، ويدعى الثاني Shift، ويمثل حالة المفاتيح Alt و Ctrl و Shift.

لتحديد المفتاح المضغوط، يمكنك مقارنة قيمة الوسيط الأول KeyCode مع ثوابت لوحة المفاتيح في فيجول بيسك، فمثلاً، إذا ضغط المستخدم على المفتاح Num Lock فإن قيمة الوسيط KeyCode سوف تساوي قيمة الثابت vbKeyNumLock، وبشكل مشابه فإن الضغط على المفتاح F1، يعني أن قيمة الوسيط KeyCode ستساوي vbKeyF1، والضغط على المفتاح 2 يعني أن قيمة الوسيط KeyCode ستساوي vbKey2، والضغط على المفتاح Home يعني أن KeyCode تساوي vbKeyHome ... وهكذا.

يعطي الجدول 12-2 لائحة بمعاني قيم الوسيط الثاني للإجراء ()cmdPushMe_KeyDown. فمثلاً، عندما تكون قيمة الوسيط الثاني تساوي 3 فهذا يعني أنه تم الضغط على كلا المفتاحين Shift و Ctrl بدون الضغط على Alt.

الجدول 12-2. القيم الممكنة للوسيط الثاني للإجراء cmdPushMe_KeyDown()

قيمة الوسيط Shift	حالة المفتاح Alt	حالة المفتاح Ctrl	حالة المفتاح Shift
0	غير مضغوط	غير مضغوط	غير مضغوط
1	غير مضغوط	غير مضغوط	مضغوط
2	غير مضغوط	مضغوط	غير مضغوط
3	غير مضغوط	مضغوط	مضغوط
4	مضغوط	غير مضغوط	غير مضغوط
5	مضغوط	غير مضغوط	مضغوط
6	مضغوط	مضغوط	غير مضغوط
7	مضغوط	مضغوط	مضغوط

يُسند نص الإجراء cmdPushMe_KeyDown() سلسلة حرفية تمثل قيمة الوسيطين السابقين إلى الخاصية Caption التابعة للافتة lblInfo.

```
lblInfo.Caption = "لقد ضغطت أي مفتاح"
lblInfo.Caption = lblInfo.Caption + vbCrLf
lblInfo.Caption = lblInfo.Caption + "KeyCode=" + _
                                     Str(KeyCode)
lblInfo.Caption = lblInfo.Caption + vbCrLf
lblInfo.Caption = lblInfo.Caption + "Shift=" + _
                                     Str(Shift)
```

يُستخدم التابع الوظيفي Str()، لتحويل القيمة الرقمية إلى قيمة نصية، يمكن وضعها في الخاصية Caption التابعة للافتة. يُحول التابع الوظيفي Str(KeyCode) الوسيط KeyCode من قيمة رقمية من النوع الصحيح إلى قيمة نصية String وكذلك يحول التابع الوظيفي Str(Shift) الوسيط Shift من قيمة رقمية إلى قيمة نصية أيضاً.

استخدم الثابت vbCrLf بغرض توزيع السلسلة الكتابية على أكثر من سطر.

التحقق من تحرير Released مفتاح ما

كما لاحظنا، يمكن استخدام وسيطي الحادثة KeyDown للتحقق من عملية ضغط أي مفتاح. تمتلك الحادثة KeyUp نفس الوسيطين المستخدمين في الحادثة KeyDown، باستثناء أن هذه الحادثة تحصل عند تحرير المفتاح من الضغط وليس عند ضغطه.

- أدخل النص التالي ضمن الإجراء cmdPushMe_KeyUp() للنموذج frmKeys حتى تتعرف على الحادثة KeyUp بشكل أوضح:

```
Private Sub cmdPushMe_KeyUp(KeyCode As Integer,_
                              Shift As Integer)
   lblInfo.Caption = "لقد حررت أي مفتاح"
   lblInfo.Caption = lblInfo.Caption + vbCrLf
   lblInfo.Caption = lblInfo.Caption + "KeyCode=" + _
                                      Str(KeyCode)
   lblInfo.Caption = lblInfo.Caption + vbCrLf
   lblInfo.Caption = lblInfo.Caption + "Shift=" + _
                                      Str(Shift)
End Sub
```

- احفظ المشروع بالطريقة المعتادة.
- نفّذ برنامج المفاتيح، وانقر الزر **اضغط هذا الزر**.

يستجيب البرنامج بإظهار مستطيل منقط حول الزر، مما يعني أن الزر يمتلك الآن تركيز لوحة المفاتيح.

- اضغط مفتاحاً ما.

يستجيب البرنامج بإظهار الرقم المرافق للمفتاح المضغوط.

- حرر المفتاح المضغوط.

يستجيب برنامج المفاتيح بإظهار قيمة المفتاح المحرر.

- انقر الزر **خروج** لإنهاء برنامج المفاتيح.

يمتلك الإجراء ()cmdPushMe_KeyUp نفس وسيطي الإجراء ()cmdPushMe_KeyDown، إلا أن الإجراء ()cmdPushMe_KeyDown يعلن عن المفتاح المضغوط، بينما يعلن الإجراء ()cmdPushMe_KeyUp عن المفتاح المحرر.

التحقق من مفتاح ذي شفرة ASCII (ASCII Key)

تُستخدم الحادثة KeyPress، للتحقق من الضغط على مفتاح ذي شفرة ASCII، اتبع الخطوات التالية للتعرف على عمل الحادثة KeyPress بشكل أوضح:

- أدخل النص التالي في الإجراء ()cmdPushMe_KeyPress للنموذج frmKeys:

```
Private Sub cmdPushMe_keyPress(KeyAscii As Integer)
    Dim Char
    Char = Chr(KeyAscii)
    lblInfo.Caption = "KeyAscii = " + Str(KeyAscii) + _
                      "Char = " + Char
End Sub
```

- ألغِ عمل عبارات الإجراء ()cmdPushMe_KeyDown، أي بوضع فاصلة علوية في بداية كل عبارة بحيث تصبح كملاحظة.
- نفّذ برنامج المفاتيح.
- انقر الزر **اضغط هذا الزر**.

يستجيب البرنامج بإظهار مستطيل منقط حول الزر، مما يشير إلى أن تركيز لوحة المفاتيح متوضع عند هذا الزر.

- اضغط على المفتاح A وأبقه مضغوطاً.

يستجيب برنامج المفاتيح بإظهار قيمة ASCII لهذا المفتاح، والتي تساوي 97 للحرف a الصغير، و65 للحرف A الكبير.

- اضغط على المفتاح F1 وأبقه مضغوطاً.

لا يُظهر برنامج المفاتيح أية قيمة في اللافتة lblInfo لأن F1 لا يمتلك شفرة ASCII، ولهذا لا تحصل الحادثة KeyPress عند الضغط على F1.

- انقر الزر **خروج** لإنهاء برنامج المفاتيح.

نص cmdPushMe_KeyPress()

يُنفذ هذا الإجراء عند الضغط على مفتاح آسكي ASCII Key، يمتلك هذا الإجراء وسيطاً واحداً فقط يمثل قيمة ASCII للمفتاح المضغوط.

يحول التابع الوظيفي Chr() القيمة KeyAscii الصحيحة إلى حرف:

```
Char = Chr(KeyAscii)
```

المتحول Char يحتفظ الآن برمز المفتاح المضغوط، فمثلاً، يحتوي Char على الرمز Z إذا ضغط المستخدم على الحرف Z، تُسند آخر عبارة في هذا الإجراء سلسلة إلى الخاصية Caption للافتة lblInfo:

```
lblInfo.Caption = "KeyAscii =" + Str(KeyAscii)+ _
                  " Char=" +Char
```

وبهذا، يتمكن المستخدم من مشاهدة رمز المفتاح المضغوط، إضافة إلى قيمة ASCII المرافقة له.

اعتراض المفاتيح بواسطة الإجراء Form_ KeyPress()

يتمكن النموذج من امتلاك تركيز لوحة المفاتيح، عندما لا يكون فيه عناصر تحكم، أو عندما تكون هذه العناصر معطّلة. ولكن في معظم البرامج، يمتلك النموذج بعض عناصر التحكم الفعّالة، ولهذا لا تُنفذ الإجراءات Form_KeyDown() و Form_KeyUp() و Form_KeyPress() أبداً.

ولإجبار البرنامج على تنفيذ هذه الإجراءات، حتى عندما لا يمتلك النموذج تركيز لوحة المفاتيح، أسند القيمة True إلى الخاصية KeyPreview للنموذج. اتبع الخطوات التالية لرؤية تأثير الخاصية KeyPreview على سير العمل:

- أسند القيمة True إلى الخاصية KeyPreview للنموذج frmKeys.
- ألغِ عمل الإجراءات cmdPushMe_KeyDown() و cmdPushMe_KeyUp() و cmdPushMe_KeyPress()، بوضع فاصلة علوية (') قبل عبارات هذه الإجراءات.
- أدخل النص التالي ضمن الإجراء Form_KeyPress():

```
Private Sub Form_keyPress(KeyAscii As Integer)
    Dim Char
    Char = Chr(KeyAscii)
    lblInfo.Caption = "KeyAscii =" + Str(KeyAscii)+ _
                      " Char=" + Char
End Sub
```

- احفظ عملك.
- نفّذ برنامج المفتاح.
- اضغط أي مفتاح آسكي.

تُظهر اللافتة lblInfo. المفتاح المضغوط بغض النظر عن عنصر التحكم الذي يمتلك تركيز لوحة المفاتيح، (طبعاً هذا بسبب إسناد القيمة True إلى الخاصية KeyPreview للنموذجfrmKeys).

تستطيع بطريقة مشابهة استخدام الإجراء Form_KeyDown()، والإجراء Form_KeyUp()، لتصيّد المفاتيح المضغوطة حتى وإن لم يمتلك النموذج تركيز لوحة المفاتيح. وعلى كل حال، نعود إلى تذكيرك بضرورة إسناد القيمة True إلى الخاصية KeyPreview للنموذج، في حال أردت تصيّد المفاتيح التي تضغط أثناء عمل البرنامج.

تُمكّن الخاصية KeyPreview البرنامج، من تصيّد حوادث لوحة المفاتيح، مما يسمح لك بكتابة نص برنامج معين، يستجيب لضغطات المفاتيح، بغض النظر عن عنصر التحكم الذي يمتلك تركيز لوحة المفاتيح.

برنامج متصيد المفاتيح

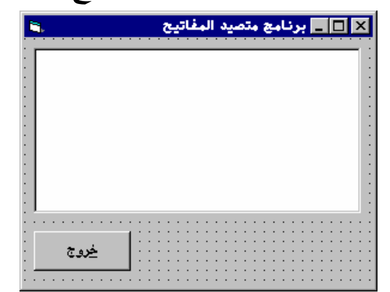
يوضّح برنامج متصيد المفاتيح، كيف يتمكن برنامجك من تصيّد مفاتيح آسكي (المفاتيح التي لها شفرات آسكي موافقة).

التمثيل المرئي لبرنامج متصيد المفاتيح

- كالعادة أنشئ مشروعاً جديداً من نوع Standard EXE، واحفظ نموذج المشروع بالاسم Upper.frm في الدليل C:\VB5Prg\Ch12 واحفظ ملف المشروع في ذات الدليل بالاسم Upper.Vbp.

- أنشئ النموذج تبعاً للجدول 12-3.

يُفترض أن يظهر النموذج المكتمل كما في الشكل 12-3.



الشكل 12-3

النموذج frmUpper

(طور التصميم).

الجدول 12-3. جدول خصائص البرنامج frmUpper.

الكائن	الخاصية	القيمة
Form	**Name**	**frmUpper**
	Caption	برنامج متصيد المفاتيح
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
TextBox	**Name**	**txtUserArea**
	ScrollBars	3-Both
	Text	(اجعله فارغاً)
	RightToLeft	True

إدخال نص برنامج متصيد المفاتيح

سنكتب الآن نص برنامج متصيد المفاتيح:

- اكتب العبارة Option Explicit في قسم التصاريح العامة General Declarations.

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- اكتب النص التالي ضمن الإجراء cmdExit_Click() للنموذج frmUpper:

```
Private Sub cmdExit_Click()
```

```
    End
End Sub
```

تنفيذ برنامج متصيد المفاتيح

- نفّذ برنامج متصيد المفاتيح.
- اكتب شيئاً ما في مربع النص.
- جرب الضغط على المفتاح Enter للانتقال إلى السطر التالي.

يرفض برنامج متصيد المفاتيح الانتقال إلى السطر التالي في مربع النص، لأن الخاصية MultiLine لمربع النص txtUserArea تساوي False حالياً. لاحظ أن الحاسب يصدر صوت رنين، كلما ضغطت المفتاح Enter، لتنبيهك أنك تضغط مفتاحاً غير مسموحٍ به.

- انقر الزر **خروج** لإنهاء برنامج متصيد المفاتيح.

تصيد المفتاح Enter

كما بيّنا، يعطي الحاسب صوت رنين، كلما ضغط المستخدم على مفتاح غير مسموح به. يمكننا تصيّد الحادثة، ومغافلة برنامج متصيد المفاتيح كما يلي:

- أدخل النص التالي ضمن الإجراء ()txtUserArea_KeyPress للنموذج frmUpper:

```
Private Sub txtUserArea_KeyPress(KeyAscii As Integer)
    'Enter إلغاء ضغطة المفتاح
    If KeyAscii = vbKeyReturn Then
      KeyAscii = 0
    End If
End Sub
```

- نفّذ برنامج متصيد المفاتيح.
- اكتب شيئاً ما في مربع النص ثم اضغط على Enter.

كما تشاهد لا يسمح لك مربع النص بالانتقال إلى السطر التالي، لكن الحاسب لا يصدر صوتاً عندما نضغط على Enter.

- أنهِ البرنامج بالنقر على الزر **خروج**.

نص الإجراء ()txtUserArea_KeyPress

ينفذ هذا الإجراء عند الضغط على مفتاح آسكي ما، يتحقق الإجراء من الضغط عل المفتاح Enter. فإذا تم الضغط على Enter، يغير الإجراء قيمة الوسيط KeyASCII إلى الصفر، فيعتقد مربع النص أنك ضغطت على المفتاح ذي قيمة الآسكي صفر (كأنك لم تضغط أي مفتاح)، وهذا هو السبب الذي يمنع الحاسب من إصدار صوت عند الضغط على Enter.

يوضح نص الإجراء ()txtUserArea_KeyPress أن هذا الإجراء، ينفذ قبل أن تسنح الفرصة لمربع النص بمعالجة المفتاح المضغوط.

تعديل برنامج متصيد المفاتيح

عدّل برنامج متصيد المفاتيح كما يلي:

- غير خانة الخاصية MultiLine لمربع النص txtUserArea إلى True.
- نفّذ برنامج متصيد المفاتيح.
- اكتب شيئاً ما في مربع النص واضغط Enter.

كما تلاحظ، لا يسمح لك بالانتقال إلى السطر التالي لأن المفتاح Enter يتم تصيده من قبل الإجراء ()txtUserArea_KeyPress، وإلغاء دوره بتغيير قيمة الوسيط KeysASCII إلى الصفر، مما يثير لدى مربع النص الاعتقاد بأنك لم تضغط شيئاً.

- أنهِ البرنامج بالطريقة المعتادة.

تحويل رموز ASCII إلى حروف كبيرة (تحويل الحرف الصغير إلى كبير)

قد يحتاج المستخدم في بعض البرامج إلى تحويل الحروف الصغيرة إلى كبيرة فقط، أو صغيرة فقط، لتوضيح كيفية إنجاز ذلك، غير نص الإجراء ()txtUserArea_KeyPress بحيث يبدو كما يلي:

```
Private Sub TxtUserArea_keyPress(KeyAscii As Integer)
    Dim Char
    Char = Chr(KeyAscii)
```

```
    KeyAscii = Asc(UCase(Char))
End Sub
```

- نفّذ برنامج متصيد المفاتيح.
- اكتب شيئاً ما ضمن مربع النص.

كما تشاهد تظهر الأحرف كأحرف كبيرة بغض النظر عن حالة المفتاحين Shift و Caps Lock.

- أنهِ البرنامج بالطريقة المعتادة.

يستخدم نص الإجراء txtUserArea_KeyPress() التابع الوظيفي Chr() لتحويل شفرة الحرف المضغوط إلى رمز أو حرف:

```
    Char = Chr(KeyAscii)
```

ثم يحوّل الرمز نفسه إلى رمز كبير (حرف كبير)، فمثلاً، القيمة المعادة من UCase("a") هي A، والقيمة المعادة من UCase("A") هي أيضاً A.

يحوّل الإجراء بعد ذلك، القيمة المعادة من التابع الوظيفي UCase()، إلى قيمة صحيحة باستخدام التابع الوظيفي Asc():

```
    KeyAscii = Asc(UCase(Char))
```

يعيد التابع الوظيفي Asc() قيمة صحيحة، تمثل قيمة الآسكي للوسيط الممرر له.

باختصار نقول، يُزوّد الإجراء txtUserArea_KeyPress() قيمة صحيحة، تمثل قيمة آسكي ASCII للمفتاح المضغوط، تحوّل هذه القيمة إلى حرف، ثم يحوّل الحرف إلى حرف كبير، ثم يُسند قيمة آسكي للحرف الكبير، إلى الوسيط KeyAscii مرة أخرى. وهكذا يعتقد مربع النص، أن المستخدم ضغط على حرف (رمز) كبير.

ملاحظة
يوجد تابع وظيفي معاكس لوظيفة التابع UCase() (**U**pper **Case**)، وهو التابع LCase() (**L**ower **Case**)، ووظيفته تحويل الأحرف إلى أحرف صغيرة دائماً.

ينبغي التنويه أنه ليس لهذه الوظائف أي تأثير على النصوص العربية، وتعيد النص كما هو بدون تغيير. فمثلاً، تعود العبارة التالية:

```
Print UCase("فـيـجـول بـيـسك")
```

بالنتيجة:

```
فـيـجـول بـيـسك
```

الخاصية Cancel

تُستخدم الخاصية Cancel لتوفير استجابة للضغط على المفتاح Esc، اتبع الخطوات التالية لرؤية دور الخاصية Cancel:

- نفّذ برنامج متصيد المفاتيح.
- اضغط المفتاح Esc من على لوحة المفاتيح.

كما تشاهد، لا يستجيب البرنامج للمفتاح Esc، بغض النظر عن موقع تركيز لوحة المفاتيح.

- انقر الزر **خروج** لإنهاء برنامج متصيد المفاتيح.
- غير حالة الخاصية Cancel للزر **خروج** إلى True.
- نفّذ برنامج متصيد المفاتيح.
- اضغط المفتاح Esc على لوحة المفاتيح.

*يستجيب البرنامج للمفتاح Esc كما لو أن المستخدم ضغط على زر الأمر **خروج**، وهذا بغض النظر عن عنصر التحكم الذي يستحوذ على تركيز لوحة المفاتيح. وهذا بسبب إسناد القيمة True إلى الخاصية Cancel لزر الأمر **خروج**.*

بشكل مشابه، يؤدي إسناد القيمة True إلى الخاصية Default للزر **خروج** إلى أثر استجابة البرنامج للضغط على المفتاح Enter وكأنه نقر على الزر **خروج** مما يتسبب بإنهاء البرنامج، وهذا بغض النظر عن أين يقع تركيز لوحة المفاتيح.

برنامج المفتاح Tab

يسمح لك ويندوز بالانتقال من عنصر تحكم لآخر، بواسطة المفتاح Tab (نقل تركيز لوحة المفاتيح من عنصر تحكم إلى آخر)، بينما يؤدي الضغط على Shift + Tab إلى الانتقال باتجاه معاكس، يوضح برنامج المفتاح Tab آلية العمل هذه.

- أنشئ مشروعاً جديداً من النوع Standard EXE، واحفظ نموذج المشروع بالاسم Tab.frm في الدليل C:\VB5Prg\Ch12 واحفظ ملف المشروع بالاسم Tab.Vbp في الدليل نفسه.
- أنشئ نموذج المشروع وفق الجدول 12-4.

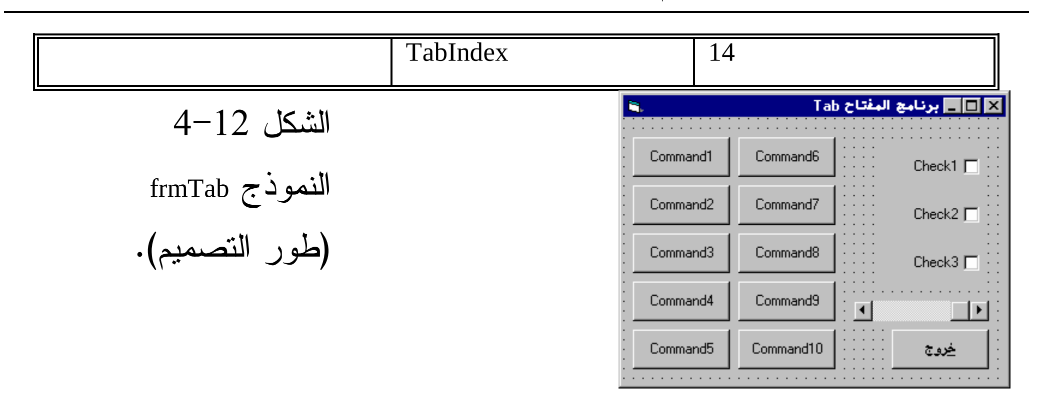
يُفترض أن يظهر النموذج المكتمل كما في الشكل 12-4.

الجدول 12-4. جدول خصائص النموذج frmTab.

الكائن	الخاصية	القيمة
Form	**Name**	**frmTab**
	Caption	برنامج المفتاح Tab
	RightToLeft	True
CommandButton	**Name**	**Command1**
	Caption	Command1
	TabIndex	0
CommandButton	**Name**	**Command2**
	Caption	Command2
	TabIndex	1
CommandButton	**Name**	**Command3**
	Caption	Command3
	TabIndex	2
CommandButton	**Name**	**Command4**
	Caption	Command4
	TabIndex	3
CommandButton	**Name**	**Command5**
	Caption	Command5

الكائن	الخاصية	القيمة
	TabIndex	4
CommandButton	**Name**	**Command6**
	Caption	Command6
	TabIndex	5
CommandButton	**Name**	**Command7**
	Caption	Command7
	TabIndex	6
CommandButton	**Name**	**Command8**
	Caption	Command8
	TabIndex	7
CommandButton	**Name**	**Command9**
	Caption	Command9
	TabIndex	8
CommandButton	**Name**	**Command10**
	Caption	Command10
	TabIndex	9
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	TabIndex	10
CheckBox	**Name**	**Check1**
	Caption	Check1
	TabIndex	11
CheckBox	**Name**	**Check2**
	Caption	Check2
	TabIndex	12
CheckBox	**Name**	**Check3**
	Caption	Check3
	TabIndex	13
Horizontal Scroll Bars	**Name**	**HScroll1**
	Max	50

14	TabIndex	



الشكل 12-4
النموذج frmTab
(طور التصميم).

إدخال نص برنامج المفتاح Tab

- تأكد من احتواء قسم التصاريح العامة على العبارة OptionExplicit:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء ()cmdExit_Click للنموذج frm:

```
Private Sub cmdExit_Click()
    End
End Sub
```

تنفيذ برنامج المفتاح Tab

- نفّذ برنامج المفتاح Tab.
- انقل تركيز لوحة المفاتيح من عنصر تحكم لآخر، بضغط المفتاح Tab، أو على مفاتيح الأسهم، لاحظ أن مفتاحي الأسهم اليميني واليساري يستخدمان لتحريك الزالقة عندما يوضع التركيز عند شريط التمرير.
- أنهِ البرنامج بنقر الزر **خروج**.

الخاصية TabIndex

تحدد الخاصية TabIndex الترتيب الذي تستقبل عناصر التحكم وفقه تركيز لوحة المفاتيح. فمثلاً، إذا كان تركيز لوحة المفاتيح حالياً عند عنصر تحكم، وقيمة الخاصية TabIndex لهذا العنصر تساوي 5، يؤدي الضغط على المفتاح Tab إلى انتقال تركيز لوحة المفاتيح إلى العنصر الذي تساوي فيه قيمة الخاصية TabIndex إلى 6، بينما إذا

كانت قيمة الخاصية TabIndex تساوي 5، فإن الضغط على Shift+Tab يؤدي إلى نقل تركيز لوحة المفاتيح إلى عنصر التحكم الذي تساوي قيمة الخاصية TabIndex فيه إلى 4.

يعمل ترتيب الانتقال وفق Tab بطريقة دائرية: فإذا كان عنصر التحكم الذي يمتلك حالياً تركيز لوحة المفاتيح يحمل أعلى قيمة للخاصية TabIndex، فإن الضغط مجدداً على Tab سيؤدي إلى انتقال تركيز لوحة المفاتيح، إلى العنصر الذي تساوي قيمة الخاصية TabIndex فيه إلى الصفر، وبالعكس عند الضغط على Shift + Tab وأنت في عنصر تحكم قيمة الخاصية TabIndex فيه تساوي الصفر، ينتقل تركيز لوحة المفاتيح إلى عنصر التحكم الذي يحمل أعلى قيمة للخاصية TabIndex.

ملاحظة

لا تتقبل بعض عناصر التحكم تركيز لوحة المفاتيح، مثالها عنصر تحكم اللافتة Label.

يُسند فيجول بيسك أرقام متسلسلة إلى الخاصية TabIndex، فهذه الخاصية لأول عنصر تساوي الصفر، وتساوي TabIndex للعنصر الذي يليه 1، وهكذا وعموماً تستطيع تغيير قيمة الخاصية TabIndex لتحديد الترتيب الذي يناسبك ضمن البرنامج.

برنامج التركيز

يستعرض برنامج التركيز، الطريقة التي تمكّن البرنامج من اكتشاف متى يمتلك عنصر التحكم أو يفقد تركيز لوحة المفاتيح.

التمثيل المرئي لبرنامج التركيز

سنبدأ كالعادة بطور التمثيل المرئي للنموذج البرنامج:

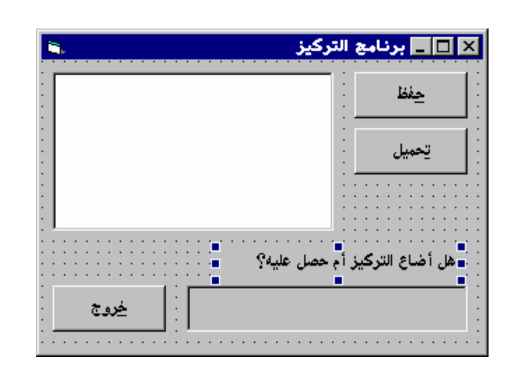
- أنشئ مشروعاً جديداً من النوع Standard EXE، واحفظ نموذج المشروع بالاسم Focus.frm في الدليل C:\VB5Prg\Ch12 واحفظ ملف المشروع بالاسم Focus.Vbp في الدليل C:\VB5Prg\Ch12.

- أنشئ النموذج طبقاً للجدول 12-5

يُفترض أن يظهر النموذج المكتمل كما في الشكل 12-5.

الجدول 12-5. جدول خصائص البرنامج frmFocus.

الكائن	الخاصية	القيمة
Form	**Name**	**frmFocus**
	Caption	برنامج التركيز
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdLoad**
	Caption	&تحميل
	RightToLeft	True
CommandButton	**Name**	**cmdSave**
	Caption	&حفظ
	RightToLeft	True
Label	**Name**	**lblInfo**
	Alignment	2-Center
	BorderStyle	1-Fixed Single
	Caption	(اجعله فارغاً)
	RightToLeft	True
Label	**Name**	**LblTitle**
	Caption	هل أضاع التركيز أم حصل عليه؟
	RightToLeft	True
TextBox	**Name**	**txtUserArea**
	MultiLine	True
	Text	(اجعله فارغاً)
	RightToLeft	True



الشكل 12-5

النموذج frmFocus

(طور التصميم).

إدخال نص برنامج التركيز

- أدخل العبارة OptionExplicit في قسم التصاريح العامة:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء cmdExit_Click():

```
Private Sub cmdExit_Click()
    End
End Sub
```

- أدخل النص التالي ضمن الإجراء txtUserArea_GotFocus():

```
Private Sub txtUserArea_GotFocus()
  lblInfo.Caption = "لقد حصل مربع النص على التركيز"
End Sub
```

- أدخل النص التالي ضمن الإجراء txtUserArea_LostFocus():

```
Private Sub txtUserArea_LostFocus()
  lblInfo.Caption = "لقد أضاع مربع النص التركيز"
End Sub
```

تنفيذ برنامج التركيز

- نفّذ برنامج التركيز.
- انقل تركيز لوحة المفاتيح إلى مربع النص المدعو txtUserArea بالنقر داخله بواسطة الفأرة، أو باستخدام المفتاح Tab أو مفاتيح الأسهم.

تُظهر اللافتة lblInfo الرسالة التالية، عند انتقال تركيز لوحة المفاتيح إلى مربع النص.

```
لقد حصل مربع النص على التركيز
```

- انقل تركيز لوحة المفاتيح بعيداً عن مربع النص txtUserArea، كأن تنقر على أحد الزرين **حفظ** أو **تحميل**.

تظهر اللافتة lblInfo الرسالة التالية، عند انتقال تركيز لوحة المفاتيح عن مربع النص.

```
لقد أضاع مربع النص التركيز
```

- أنهِ البرنامج بنقر الزر **خروج**.

كيف يعمل برنامج التركيز

يستخدم البرنامج الحادثتين GotFocus (حصل على التركيز) و LostFocus (أضاع التركيز)، لاكتشاف ما إذا كان مربع النص قد استحوذ أو خسر تركيز لوحة المفاتيح.

نص الإجراء ()txtUserArea_GotFocus

يُنفذ هذا الإجراء عند انتقال تركيز لوحة المفاتيح إلى مربع النص، ويُسند رسالة إلى الخاصية Caption للافتة lblInfo مضمونها، أن مربع النص يستحوذ الآن على تركيز لوحة المفاتيح:

```
Private Sub txtUserArea_GotFocus()
   lblInfo.Caption = "لقد حصل مربع النص على التركيز"
End Sub
```

نص الإجراء ()txtUserArea_LostFocus

يُنفذ هذا الإجراء عند ضياع تركيز لوحة المفاتيح من مربع النص، ويُسند رسالة إلى الخاصية Caption للافتة lblInfo مضمونها، أن مربع النص فقد تركيز لوحة المفاتيح:

```
Private Sub txtUserArea_LostFocus()
   lblInfo.Caption = "لقد أضاع مربع النص التركيز"
End Sub
```

الخلاصة

تعلمنا في هذا الفصل، كيفية تحديد المفتاح المضغوط باستخدام الحوادث KeyDown و KeyUp و KeyPress، وتعلمنا أنه عندما تكون قيمة KeyPreview مساوية إلى True، فإن الإجراءات ()Form_KeyDown و ()Form_KeyUp و ()Form_KeyPress، سوف تُنفذ، بغض النظر عن عنصر التحكم الذي يمتلك تركيز لوحة المفاتيح.

كما تعلمنا أيضاً دور الخاصية TabIndex وأثرها على ترتيب إنتقال تركيز لوحة المفاتيح جيئة وذهاباً، عند ضغط المفتاح Tab وشاهدنا أيضاً كيف تحصل الحادثتان GotFocus و LostFocus عندما يستحوذ عنصر تحكم ما على تركيز لوحة المفاتيح أو يفقده.

الفصل الثالث عشر

عناصر تحكم نظام الملفات

يركّز هذا الفصل على استخدام عناصر تحكم نظام الملفات، لكتابة برنامج يمكّن المستخدم من اختيار ملف ما من أي محرك أقراص.

توجد ثلاثة أنواع من عناصر التحكم هذه، وهي:

- مربع سرد الأدلة Drive List Box.
- مربع سرد الملفات File List Box.
- مربع سرد السواقات Directory List Box.

تُستخدم الأنواع الثلاثة هذه، جنباً إلى جنب في برنامج نموذجي، يسمح للمستخدم باختيار الملفات من محركات الأقراص المختلفة. ليظهر مربع حوار مخصص، عندما يرغب المستخدم باختيار ملف ما، ويحوي هذا المربع على عناصر التحكم الثلاثة هذه (انظر الشكل 13-1).

يمكنك بعدها اختيار الملف المطلوب، بانتقاء محرك الأقراص المناسب من مربع سرد محركات الأقراص، ثم انتقاء دليل معين من مربع سرد الأدلة، وأخيراً اختيار ملف من مربع سرد الملفات.



الشكل 13-1
عناصر تحكم نظام الملفات الثلاث.

ملاحظة

تعلمنا من الفصل السادس، كيفية استخدام عنصر التحكم Common Dialog لإظهار مربع الحوار **فتح**، أو مربع الحوار **حفظ**. وكما رأينا من ذلك الفصل، فإن عملية إضافة عنصر تحكم مربعات الحوار الشائعة في البرامج بالغة السهولة.

إذاً ما هو الدافع إلى تعلم كيفية استخدام عناصر تحكم نظام الملفات ؟!.

سيتبين لك من هذا الفصل، أنها تعطي قدرة أكبر على التحكم، مما تعطيه مربعات الحوار الشائعة.

برنامج الحجم

سنكتب برنامجاً ندعوه برنامج الحجم، ويتضمن ثلاثة عناصر تحكم نظام الملفات، وتستطيع استخدامه لاختيار ملف ما من محرك أقراص، وإظهار حجم الملف المنتقى.

يُفترض في برنامج الحجم إنجاز ما يلي:

- إظهار نموذج على الشاشة لاختيار الملفات عند تشغيل البرنامج، حسب ما يبينه الشكل13-2. يحتوي إطار برنامج الحجم، على مربع تحرير وسرد يدعى

نوع الملف، ويقع تحت مربع سرد الملفات، ويمكّنك من اختيار نوع ملف محدد من لائحة أنواع الملفات الجاهزة.

- إظهار لائحة محركات الأقراص المتوافرة، وتمكين المستخدم الاختيار من بينها.
- إظهار لائحة الملفات الموجودة حالياً، في الدليل الذي يختاره المستخدم من مربع سرد الأدلة.
- إظهار اسم الملف في مربع النص **اسم الملف** لدى اختياره من مربع لائحة الملفات.
- إظهار حجم الملف المنتقى عند نقر الزر **موافق**. (انظر الشكل 13-3).



الشكل 13-2

برنامج الحجم.



الشكل 13-3

إظهار حجم الملف المنتقى.

- إنهاء تنفيذ البرنامج لدى الضغط على المفتاح Esc.
- إظهار الملفات المنتمية إلى النوع الذي تختاره من مربع التحرير والسرد **(نوع الملف)** فقط، فمثلاً عند اختيار الملفات النصية (*.TXT)، تظهر الملفات ذات الامتداد TXT فقط في مربع سرد الملفات.

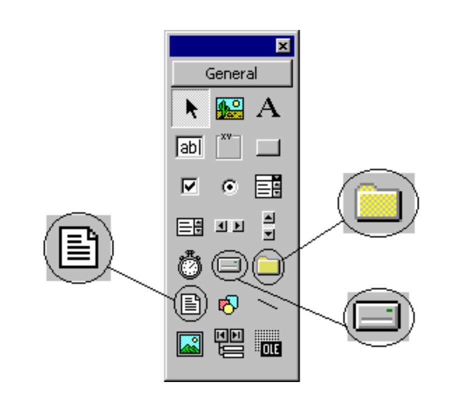
التمثيل المرئي لبرنامج الحجم

سنبدأ كعادتنا بطور التمثيل المرئي لنموذج البرنامج:

- أنشئ الدليل C:\VB5Prg\Ch13 لأننا سنستعمله لحفظ العمل المنجز.
- أنشئ مشروعاً جديداً من النوع Standard EXE.
- احفظ نموذج المشروع بالاسم Size.Frm في الدليل C:\VB5Prg\Ch13 واحفظ ملف المشروع بالاسم Size.Vbp في الدليل C:\VB5Prg\Ch13.
- أنشئ النموذج طبقاً للجدول 13-1.

يُفترض أن يبدو النموذج المكتمل كما في الشكل13-2.

ملاحظة
يوضح الشكل 13-4 مواقع الرموز التي تمثل عناصر تحكم نظام الملفات الثلاث، في إطار مربع الأدوات. طبعاً قد تختلف هذه المواقع عمّا هو عليه في شريط الأدوات لدى المستخدم.



الشكل 13-4

رموز عناصر تحكم نظام الملف في إطار مربع الأدوات.

الجدول 13–1. جدول خصائص برنامج الحجم.

الكائن	الخاصية	القيمة
Form	**Name**	**frmSize**
	Caption	برنامج الحجم
	RightToLeft	True
CommandButton	**Name**	**cmdOk**
	Caption	&موافق
	RightToLeft	True
CommandButton	**Name**	**cmdCancel**
	Caption	إل&غاء
	RightToLeft	True
Directory List Box	**Name**	**dirDirectory**
Drive List Box	**Name**	**drvDrive**
File List Box	**Name**	**filFiles**
TextBox	**Name**	**txtFileName**
	Text	(اجعله فارغاً)
ComboBox	**Name**	**cboFileType**
	Style	2-Dropdown List
Label	**Name**	**lblFileName**
	Caption	اسم الملف:
	RightToLeft	True
Label	**Name**	**lblFileType**
	Caption	نوع الملف:
	RightToLeft	True
Label	**Name**	**lblDirectory**
	Caption	الأدلة:
	RightToLeft	True
Label	**Name**	**LblDirName**

الكائن	الخاصية	القيمة
	Caption	(اجعله فارغاً)
	BorderStyle	1-Fixed Single
	RightToLeft	True
Label	**Name**	**lblDrive**
	Caption	السواقة:
	RightToLeft	True

إدخال نص برنامج الحجم

- أدخل العبارة Option Explicit في قسم التصاريح العامة للنموذج frmSize:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- اكتب النص التالي ضمن الإجراء ()Form_Load:

```
Private Sub Form_Load()
    cboFileType.AddItem "All files (*.*)"
    cboFileType.AddItem "Text files(*.TXT)"
    cboFileType.AddItem "Doc files (*.DOC)"
    cboFileType.ListIndex = 0
    lblDirName.Caption = dirDirectory.Path
End Sub
```

- اكتب النص التالي ضمن الإجراء ()drvDrive_Change:

```
Private Sub drvDrive_Change()
    On Error GoTo DriveError
    dirDirectory.Path = drvDrive.Drive
    Exit Sub
    DriveError:
      MsgBox "Drive Error !", vbExclamation, "Error"
      drvDrive.Drive = dirDirectory.Path
      Exit Sub
End Sub
```

- اكتب النص التالي ضمن الإجراء ()dirDirectory_Change:

```
Private Sub dirDirectory_Change()
    filFiles.Path = dirDirectory.Path
```

```
    lblDirName.Caption = dirDirectory.Path
End Sub
```

❑ اكتب النص التالي ضمن الإجراء ()cboFileType_Click:

```
Private Sub cboFileType_Click()
    Select Case cboFileType.ListIndex
    Case 0
        filFiles.Pattern = "*.*"
    Case 1
        filFiles.Pattern = "*.TXT"
    Case 2
        filFiles.Pattern = "*.DOC"
    End Select
End Sub
```

❑ أدخل النص التالي ضمن الإجراء ()filFiles_Click:

```
Private Sub filFiles_Click()
    txtFileName.Text = filFiles.filename
End Sub
```

❑ أدخل النص التالي ضمن الإجراء ()cmdOk_Click:

```
Private Sub cmdOk_Click()
    Dim PathAndName As String
    Dim FileSize As String
    Dim Path As String
    '
    If txtFileName.Text = "" Then
        MsgBox "يجب أن تختار ملفاً!", _
        vbMsgBoxRight Or vbMsgBoxRtlReading, "برنامج الحجم"
        Exit Sub
    End If
    If Right(filFiles.Path, 1) <> "\" Then
      Path = filFiles.Path + "\"
    Else
      Path = filFiles.Path
    End If

    If txtFileName.Text = filFiles.filename Then
      PathAndName = Path + filFiles.filename
    Else
```

```
  PathAndName = txtFileName.Text
End If
On Error GoTo FileLenError
'
FileSize = Str(FileLen(PathAndName))
MsgBox "حجم الملف " + PathAndName + ":هو " _
 + FileSize + " بايت",vbMsgBoxRight Or vbMsgBoxRtlReading,_
 "برنامج الحجم"
Exit Sub
FileLenError:
MsgBox "لم أستطع إيجاد حجم الملف " + PathAndName, _
  vbMsgBoxRight Or vbMsgBoxRtlReading, _
  "برنامج الحجم"
Exit Sub
End Sub
```

❑ أدخل النص التالي ضمن الإجراء filFiles_DblClick():

```
Private Sub filFiles_DblClick()
    txtFileName.Text = filFiles.filename
    cmdOk_Click
End Sub
```

❑ أدخل النص التالي ضمن الإجراء cmdCancel():

```
Private Sub cmdCancel_Click()
    End
End Sub
```

❑ احفظ المشروع باختيار **Save Project** من القائمة **File** لفيجول بيسك.

تنفيذ برنامج الحجم

لنشاهد عمل برنامج الحجم:

❑ نفّذ برنامج الحجم وتمرن على شتى عناصر التحكم التي تظهر على الشاشة.

لاحظ المظاهر التالية عند تشغيلك لبرنامج الحجم:

- تظهر أدلة محرك الأقراص الذي تنتقيه، في مربع سرد الأدلة، حاول اختيار محرك أقراص من مربع سرد محركات الأقراص.
- عند اختيار محرك أقراص غير جاهز، تظهر رسالة خطأ، ويسترجع مربع

السرد قيمته الأصلية. فمثلاً لدى محاولتك القراءة من محرك الأقراص A ولم يكن هنالك قرص مرن، تظهر رسالة خطأ، ويسترجع مربع سرد المحركات قيمته الافتراضية (وهي:C).

■ يؤدي النقر المزدوج على الدليل المطلوب في مربع سرد الأدلة إلى اختياره.

■ تظهر حال اختيار أحد الأدلة، ملفات ذلك الدليل في مربع سرد الملفات، ويظهر اسم الدليل الحالي فوق مربع سرد الأدلة.

■ يظهر اسم الملف في مربع النص المدعو **اسم الملف** حال توضع الإضاءة فوقه، في مربع سرد الملفات.

■ بعد اختيار نوع ملف مختلف، من المربع **نوع الملف**، يُظهر مربع سرد الملفات فقط الملفات التي تنتمي للنوع الذي اخترته. لاحظ أنك لا تستطيع الكتابة في منطقة النص لمربع التحرير والسرد، وذلك بسبب إسناد القيمة DropDown إلى الخاصية Style لمربع التحرير والسرد أثناء طور التصميم.

■ تظهر عند نقر الزر **موافق** رسالة تُعطي حجم الملف المنتقى.

■يمكنك كتابة اسم الملف، في مربع النص **اسم الملف** بدلاً من انتقائه من مربع سرد الملفات.

■ يؤدي ضغط المفتاح Enter إلى نفس دور نقر الزر **موافق** وذلك بسبب إسناد القيمة True إلى الخاصية Default للزر **موافق** خلال زمن التصميم.

■ يؤدي ضغط المفتاح Esc إلى نفس دور نقر الزر **إلغاء** الذي أُسندَت القيمة True إليه أثناء زمن التصميم.

❑ انقر الزر **إلغاء** لإنهاء برنامج الحجم.

كيف يعمل برنامج الحجم

تمتلك عناصر تحكم نظام الملفات – كباقى عناصر التحكم الأخرى– حوادث وخصائص. ويحدد نص إجراءات الحوادث المختلفة لها، كيف تتخاطب عناصر التحكم مع بعضها.

نص الإجراء Form_Load()

يُنفذ الإجراء Form_Load() آلياً عند تشغيل البرنامج. ويتم في هذا الإجراء تجهيز مربع التحرير والسرد cboFileType واللافتة lblDirName:

```
Private Sub Form_Load()
    cboFileType.AddItem "All files (*.*)"
    cboFileType.AddItem "Text files(*.TXT)"
    cboFileType.AddItem "Doc files (*.DOC)"
    cboFileType.ListIndex = 0
    lblDirName.Caption = dirDirectory.Path
End Sub
```

تُستخدم الطريقة AddItem ثلاث مرات، لملء مربع cboFileType بثلاثة عناصر:

```
All files (*.*), Text files (*.TXT), Doc files (*.DOC)
```

إسناد القيمة صفر إلى الخاصية ListIndex لمربع نوع الملفات cboFileType، يجعلها تشير إلى أول بند من بنوده، ألا وهو البند (*.*) All Files.

وأخيراً إسناد القيمة الابتدائية للخاصية Path لمربع سرد الأدلة، إلى الخاصية Caption للافتة lblDirName، وهي الدليل الحالي، وهكذا منذ بدء تشغيل البرنامج، تُظهر اللافتة LblDirName اسم الدليل الحالي.

نص الإجراء drvDrive_Change()

يُنفّذ الإجراء drvDrive_Change() آلياً، عند تغيير محرك الأقراص في مربع سرد المحركات. يجدّد هذا الإجراء الخاصية Path لمربع سرد الأدلة، بإسناد محرك الأقراص الجديد الذي تم اختياره إليها:

```
Private Sub drvDrive_Change()
    On Error GoTo DriveError
```

```
    dirDirectory.Path = drvDrive.Drive
    Exit Sub
    DriveError:
     MsgBox "Drive Error !", vbExclamation, "Error"
     drvDrive.Drive = dirDirectory.Path
     Exit Sub
End Sub
```

قبل أن يغيّر الإجراء، الخاصية Path لمربع سرد الأدلة، كتبنا سطر مصيدة الأخطاء. تُعتبر مصيدة الأخطاء هذه لازمة، لأن تغيير مسار Path مربع سرد الأدلة، قد ينجم عنه خطأ ما.

فمثلاً، قد يغير المستخدم مربع سرد السواقات، إلى محرك الأقراص :A، إلا أن :A ليس جاهزاً في تلك اللحظة. فيتسبب تغيير المسار Path في مربع سرد الأدلة إلى :A بظهور خطأ. ولتلافي حصول خطأ أثناء زمن التنفيذ، وُضعت المصيدة الممثلة بالعبارة التالية:

```
On Error Go To DriveError
```

فإذا ظهر خطأ ما الآن أثناء تنفيذ هذه العبارة:

```
dirDirectory.Path = drvDrive.Drive
```

فسيعمل فيجول بيسك على نقل تنفيذ البرنامج إلى نص البرنامج الواقع تحت اللافتة :DriveError. ليُظهر نص البرنامج الوارد تحت هذه اللافتة، رسالة خطأ ويسترجع القيمة الأصلية لمحرك الأقراص، وذلك باستخدام العبارة التالية:

```
drvDrive.Drive = dirDirectory.Path
```

لاحظ أن قيمة dirDirectory.Path لم تتغير (أي ما زالت محافظة على قيمتها الأصلية)، لأن العبارة التي تسببت في الخطأ (في حال حدوثه) لم تنفذ بعد.

لا يحصل خطأ، إذا كان محرك الأقراص المنتقى جاهزاً، وبالتالي يتغير مسار Path مربع سرد الأدلة، إلى محرك الأقراص المنتقى، ونتيجة ذلك، يستعرض مربع سرد الأدلة، الأدلة الموجودة في محرك الأقراص الذي انتقيته.

نص الإجراء dirDirectory_Change()

يُنفذ الإجراء dirDirectory_Change()، عند تبديل الدليل الحالي في مربع سرد الأدلة. يُحدّث نص هذا الإجراء، الخاصية Path لمربع سرد الملفات، ويُسند الدليل الجديد إلى الخاصية Caption للافتة lblDirName:

```
Private Sub dirDirectory_Change()
    filFiles.Path = dirDirectory.Path
    lblDirName.Caption = dirDirectory.Path
End Sub
```

يُظهر مربع سرد الملفات، الملفات المحتواة في الدليل الذي اختاره المستخدم، نتيجة لإسناد الخاصية Path لمربع سرد الأدلة، إلى الخاصية Path لمربع سرد الملفات.

نص الإجراء cboFileType_Click()

يُنفذ هذا الإجراء، نتيجة اختيار نوع ملفات آخر، من مربع **نوع الملفات** cboFileType، يُحدّث هذا الإجراء الخاصية Pattern لمربع سرد الملفات تبعاً لنوع الملف المنتقى:

```
Private Sub cboFileType_Click()
    Select Case cboFileType.ListIndex
    Case 0
        filFiles.Pattern = "*.*"
    Case 1
        filFiles.Pattern = "*.TXT"
    Case 2
        filFiles.Pattern = "*.DOC"
    End Select
End Sub
```

تُستخدم العبارة الشرطية Select Case لتحديد نوع الملف الذي اخترته من مربع سرد أنواع الملفات cboFileType. لنفترض أن الإجراء Form_Load() ملأ المربع cboFileType بثلاثة بنود هي:

```
All Files (*.*) , Text Files(*.TXT) , Doc Files(*.DOC)
```

تُنفذ عبارة Case الموافقة للبند الذي اخترته من مربع **نوع الملفات**، فمثلاً لدى اختيار البند الثاني Text Files (*.TXT) تنفذ العبارة التالية:

```
filFiles.Pattern = "*.TXT"
```

ونتيجة لذلك، يُظهر مربع سرد الملفات، فقط الملفات ذات الامتداد TXT.

نص الإجراء ()filFiles_Click

يُنفذ الإجراء ()filFiles_Click عند اختيار ملف ما من الملفات الموجودة في مربع سرد الملفات. يُحدّث نص هذا الإجراء، مربع النص txtFileName باسم الملف المنتقى:

```
Private Sub filFiles_Click()
    txtFileName.Text = filFiles.filename
End Sub
```

نص الإجراء ()cmdOk_Click

يُنفذ الإجراء ()cmdOk_Click عند نقر الزر **موافق**، ويُظهر نص هذا الإجراء حجم الملف الذي تم اختياره للتو:

```
Private Sub cmdOk_Click()
    Dim PathAndName As String
    Dim FileSize As String
    Dim Path As String
    'عند عدم اختيار ملف أخبر المستخدم وأنه هذا الإجراء
    If txtFileName.Text = "" Then
        MsgBox "!يجب أن تختار ملفاً", _
        vbMsgBoxRight Or vbMsgBoxRtlReading, _
      "برنامج الحجم"
        Exit Sub
    End If
    'تحقق من أن المسار ينتهي بالرمز \
    If Right(filFiles.Path, 1) <> "\" Then
      Path = filFiles.Path + "\"
    Else
      Path = filFiles.Path
    End If

    If txtFileName.Text = filFiles.filename Then
      PathAndName = Path + filFiles.filename
    Else
      PathAndName = txtFileName.Text
```

```
End If
On Error GoTo FileLenError
'
FileSize = Str(FileLen(PathAndName))
MsgBox "حجم الملف " + PathAndName + ":هو " _
  + FileSize + " بايت",vbMsgBoxRight Or vbMsgBoxRtlReading,_
  "برنامج الحجم"
Exit Sub
FileLenError:
MsgBox "لم أستطع إيجاد حجم الملف " + PathAndName, _
   vbMsgBoxRight Or vbMsgBoxRtlReading, "برنامج الحجم"
Exit Sub
End Sub
```

أول شيء يفعله الإجراء، هو التأكد من أنك اخترت ملفاً ما، وذلك بمقارنة الخاصية Text لمربع النص txtFileName مع رمز الفراغ (""). فإذا تحقق الشرط تظهر رسالة مفادها أنك لم تختر ملفاً ويتم إنهاء الإجراء:

```
If txtFileName.Text = "" Then
    MsgBox "يجب أن تختار ملفاً!", _
    vbMsgBoxRight Or vbMsgBoxRtlReading, _
  "برنامج الحجم"
    Exit Sub
End If
```

أما إذا تأكد الإجراء من أن المستخدم انتقى ملفاً، فيتم عند ذلك تحديث المتحول Path بإسناد مسار الملف المنتقى إليه.

يُستخدم التابع الوظيفي Right() للتأكد بأن الرمز الواقع أقصى يمين المسار المنتقى، هو الرمز "\"، فإذا لم يكن كذلك فإنه يضيفه إلى المتحول Path.

```
If Right(filFiles.Path, 1) <> "\" Then
  Path = filFiles.Path + "\"
Else
  Path = filFiles.Path
End If
```

وبعد أن يصبح المتحول Path جاهزاً، يغدو بالوسع تحديث قيمة المتحول PathAndName بواسطة عبارة If شرطية:

```
If txtFileName.Text = filFiles.filename Then
  PathAndName = Path + filFiles.filename
Else
  PathAndName = txtFileName.Text
End If
```

تتحقق هذه العبارة، من تطابق الاسم المضاء في مربع سرد الملفات، مع الاسم الموجود في مربع النص txtFileName.

فإذا كان هنالك اختلاف، فهذا يعني أنك كتبت يدوياً مسار واسم الملف، لهذا يتم إسناد الشيء الذي كتبته إلى المتحول PathAndName. أما إذا تطابق الاسمان، فعندها تُسند السلسلة التالية إلى المتحول PathAndName:

```
Path + filFile.filename
```

وبعد تجهيز المتحول PathAndName، يصبِح بالوسع استخدام التابع الوظيفي FileLen() لإيجاد حجم الملف، وباعتبار أن استخدام التابع الوظيفي FileLen() قد يؤدي إلى حدوث خطأ أثناء زمن التنفيذ (كأن تدخل اسم ملف غير موجود)، فقد وُضعَت مصيدة للخطأ بواسطة العبارة التالية:

```
On Error GoTo FileLenError
```

فإذا وقع خطأ أثناء تنفيذ العبارة التالية:

```
FileSize = Str(FileLen(PathAndName))
```

ينتقل التنفيذ إلى العبارة الواقعة تحت اللافتة :FileLenError، والتي تُظهر بدورها رسالة خطأ، ويتم بعدها إنهاء البرنامج:

```
FileLenError:
MsgBox "لم أستطع إيجاد حجم الملف " + PathAndName, _
   vbMsgBoxRight Or vbMsgBoxRtlReading, "برنامج الحجم"
Exit Sub
```

أما إذا لم يتسبب التابع الوظيفي FileLen() بوقوع خطأ، فعند ذلك يتم إظهار حجم الملف على الشاشة وإنهاء الإجراء:

```
MsgBox "حجم الملف " + PathAndName + ":هو " _
  + FileSize + " بايت",vbMsgBoxRight Or vbMsgBoxRtlReading,_
  "برنامج الحجم"
Exit Sub
```

نص الإجراء ()filFiles_DblClick

يُنفذ هذا الإجراء، عند النقر المزدوج على أحد الملفات الموجودة في مربع سرد الملفات. يُسند نص هذا الإجراء، اسم الملف الذي تم النقر المزدوج عليه، إلى مربع النص txtFileName ويُنفذ الإجراء ()cmdOk_Click:

```
Private Sub filFiles_DblClick()
    txtFileName.Text = filFiles.filename
    cmdOk_Click
End Sub
```

نص الإجراء ()cmdCancel_Click

يُنفذ هذا الإجراء عند نقر الزر **إلغاء**، وينهي تنفيذ البرنامج:

```
Private Sub cmdCancel_Click()
    End
End Sub
```

سمات مربع لائحة الملفات

يمكن لأي ملف امتلاك أي من السمات الأربعة التالية:

- Read Only للقراءة فقط: الملف الذي يحمل هذه السمة، معد للقراءة فقط ولا يمكن تعديله أو حذفه أو الكتابة فيه ما لم يتم إزالة هذه السمة عنه.
- Hidden مخفي: لا يتمكن أمر نظام التشغيل DOS المدعو Dir، من إظهار الملفات التي تحمل هذه السمة.
- System تابع للنظام: تمتلك ملفات نظام التشغيل DOS هذه السمة، وهذه الملفات لا يمكن حذفها أو تعديلها بالكتابة عليها، ما لم تزال عنها هذه السمة.
- Archive تمت أرشفته: تُوضع هذه السمة على الملفات التي أجري عليها نسخ احتياطي بواسطة الأمر BackUp (أو أي من خدمات النسخ الاحتياطي الأخرى). وتستخدم هذه السمة كراية، للدلالة على أن الملف تم نسخه احتياطياً. يتم إزالة هذه السمة آلياً من قبل النظام، عند أي تعديل للملف، للدلالة على أنه قد

أجري عليه تعديل، ويحتاج إلى إجراء نسخ احتياطي مرة ثانية.

تُحددِ السمات Attribute لمربع سرد الملفات، أي الملفات سيتم إظهارها في مربع سرد الملفات، وذلك بالاعتماد على سمات الملفات. علماً أن السمات لمربع سرد الملفات هي Read Only و Archive و Normal و System و Hidden، ويمكن أن تحمل كل منها إحدى القيمتين True أو False. فمثلاً لإظهار الملفات المعدة للقراءة فقط، ستحتاج إلى تكليف سمات مربع سرد الملفات المدعو filMyFiles كما يلي:

```
filMyFiles.ReadOnly = True
filMyFiles.Archive = False
filMyFiles.Normal = False
filMyFiles.System = False
filMyFiles.Hidden = False
```

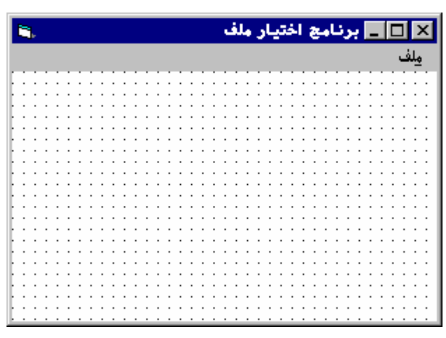
برنامج اختيار ملف

أوضح برنامج الحجم كيفية بناء نموذج يسمح لك باختيار ملف من محرك أقراص. وباعتبار أن الكثير من البرامج تحتاج إلى مثل هذا النموذج، فإنها لفكرة حسنة أن نبني نموذجاً لمربع حوار عام الغرض، يُنجز عملية اختيار الملفات من محركات الأقراص والأدلة الفرعية.

يوضح برنامج اختيار ملف، كيفية بناء واستخدام مثل هذا النموذج والذي ندعوه هنا **اختيار ملف:**

قبل المباشرة بكتابة برنامج اختيار ملف، لابد لنا من تحديد الأعمال التي يتوجب عليه إنجازها:

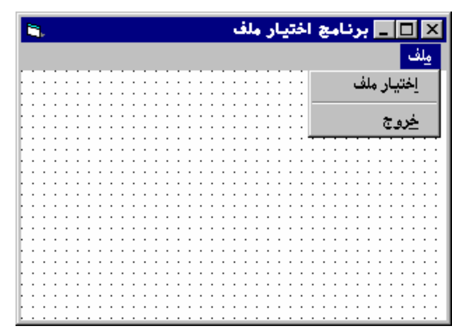
- عند تنفيذ البرنامج، يجب أن يظهر شريط قوائم يحمل القائمة ذات العنوان **ملف** (انظر الشكل 13–5).



الشكل 13-5

برنامج اختيار ملف.

■تملك القائمة **ملف** بندين هما: **اختيار ملف** و **خروج** (انظر الشكل 13-6).



الشكل 13-6

بنود القائمة **ملف**.

■يظهر مربع الحوار **اختيار ملف** عند اختيار بند القائمة **اختيار ملف** من القائمة **ملف** (الشكل 13-7). يُغلق مربع الحوار بعد اختيار ملف ويُظهر الملف المنتقى.

■ يتم إنهاء تنفيذ البرنامج، باختيار **خروج** من القائمة **ملف**.



الشكل 13-7

مربع الحوار **اختيار ملف**.

التمثيل المرئي لبرنامج اختيار ملف

سنبدأ كعادتنا بطور التمثيل المرئي لنموذج برنامج اختيار ملف:

- أنشئ مشروعاً جديداً من النوع Standard EXE.
- احفظ نموذج المشروع بالاسم Select.Frm في الدليل C:\VB5Prg\Ch13، واحفظ ملف المشروع بالاسم Select.Vbp في الدليل ذاته.
- أنشئ النموذج طبقاً للجدولين13–2 و 13–3.

يُفترض أن يظهر النموذج المكتمل كما في الشكل13–5.

الجدول 13–2. جدول خصائص برنامج اختيار ملف.

الكائن	الخاصية	القيمة
Form	**Name**	**frmSelect**
	Caption	برنامج اختيار ملف
	RightToLeft	True
Menu	(انظر الجدول 13–3)	(انظر الجدول 13–3)

الجدول 13–3. جدول قائمة النموذج frmSelect.

الاسم	العنوان
mnuFile	ملف
mnuSelectFile	...&اختيار ملف
mnuSep1	...–
mnuExit	...&خروج

ينبغي علينا الآن، تشكيل نموذج آخر وتسميته GetFile.Frm، لأن هذا النموذج سيُستخدم كمربع حوار عام الغرض، ويدعى **اختيار ملف**.

يُعتبر النموذج GetFile.Frm متطابقاً إلى درجة كبيرة مع النموذج Size.Frm، لهذا فبدلاً من إعادة كتابته، سوف ننسخ الملف Size.Frm إلى نفس الدليل الذي يقع فيه الملف

Select.frm، وسنغير الاسم من Size.frm إلى GetFile.frm، ثم سنضيفه (أي GetFile.Frm) إلى المشروع Select.Vbp.

إليك الآن طريقة إنجاز ذلك:

- استخدم مستكشف ويندوز لتشكيل نسخة أخرى عن الملف Size.frm، وسمها بالاسم GetFile.frm.

يُفترض أن يحوي الدليل C:\VB5Prg\Ch13 الآن، الملفين Size.frm و GetFile.frm.

- أضف الملف GetFile.Frm إلى المشروع Select.Vbp باتباع الخطوات التالية:
- ارجع إلى فيجول بيسك.
- اختر **Add Form** من القائمة **Project**.

يستجيب فيجول بيسك بإظهار مربع الحوار Add Form.

- اختر الصفحة Existing في مربع الحوار Add Form، (لإنا نرغب بإضافة نموذج موجود إلى المشروع).
- استخدم الصفحة Existing لمربع الحوار Add Form لاختيار الملف GetFile.frm من الدليل C:\VB5Prg\Ch13 ثم انقر الزر **فتح**.

يستجيب فيجول بيسك بإضافة الملف GetFile.frm إلى المشروع Select.Vbp ويمكنك إظهار الإطار Project والتحقق من أن المشروع Select.VBP يحوي الآن نموذجين هما:

- Select.frm.
- GetFile.frm.

حتى هذه النقطة، يعتبر النموذج GetFile.Frm متطابق تماماً مع النموذج.

يتوجب عليك تغيير الخاصية Name والخاصية Caption للنموذج GetFile.Frm كما يلي:

- اختر النموذج GetFile.frm بوضع الإضاءة على البند GetFile.frm في إطار المشروع، ثم انقر الرمز View Object (يوجد رمز هذا الزر في الزاوية اليسرى العليا من إطار المشروع).

- غيّر الخاصية Name للنموذج GetFile.frm من frmSize إلى frmGetFile.
- غيّر الخاصية Caption للنموذج GetFile.frm من **برنامج الحجم**، إلى **اختيار ملف**.

انتهى الآن طور التمثيل المرئي للنموذج frmGetFile. ويُفترض أن يظهر هذا النموذج كما في الشكل 13-7.

إدخال نص البرنامج اختيار ملف

يمتلك برنامج اختيار ملف الآن نموذجين: frmSelect و frmGetFile. سندخل عبر الفقرات التالية نص برنامج إجراءات هذين النموذجين. (لاحظ أن نصوص وإجراءات النموذج frmGetFile جاهزة مسبقاً، لأننا كتبناها أثناء تصميم النموذج frmSize).

إدخال نص النموذج frmSelect

- أدخل النص التالي في قسم التصاريح العامة للنموذج frmSelect:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي في الإجراء ()Form_Load للنموذج frmSelect:

```
Private Sub Form_Load()
    Load frmGetFile
    frmGetFile.cboFileType.AddItem "All files(*.*)"
    frmGetFile.cboFileType.AddItem "Text files(*.TXT)"
    frmGetFile.cboFileType.AddItem "Doc files(*.DOC)"
    frmGetFile.cboFileType.ListIndex = 0
End Sub
```

- أدخل النص التالي ضمن الإجراء ()mnuSelectFile_Click:

```
Private Sub mnuSelectFile_Click()
    frmGetFile.Caption = "اختيار ملف"
    frmGetFile.Show 1
    If frmGetFile.Tag = "" Then
      MsgBox "لم تختر أي ملف"
    Else
      MsgBox "لقد اخترت الملف" + frmGetFile.Tag
```

```
    End If
End Sub
```

▫ أدخل النص التالي في الإجراء mnuExit_Click() للنموذج frmSelect:

```
Private Sub mnuExit_Click()
    End
End Sub
```

إدخال نص النموذج frmGetFile

اتفقنا أن هذا النموذج مكتوب ومكتمل، فقد نسخناه عن نموذج المثال السابق (أي النموذج frmSize) ولا يلزمه سوى بعض التعديلات والإضافات.

▫أدخل النص التالي في الإجراء Form_Load() للنموذج frmGetFile بحيث يبدو كالتالي:

```
Private Sub Form_Load()
      LblDirName.Caption = dirDirectory.Path
End Sub
```

▫سنحافظ على نص الإجراء drvDrive_Change() على حاله، لهذا تأكد فقط أنه مطابق لما يلي:

```
Private Sub drvDrive_Change()
     On Error GoTo DriveError
     dirDirectory.Path = drvDrive.Drive
      Exit Sub
    DriveError:
     MsgBox "Drive Error !", vbExclamation, "Error"
     drvDrive.Drive = dirDirectory.Path
     Exit Sub
End Sub
```

▫سنحافظ على نص الإجراء dirDirectory_Click() على حاله، لهذا تأكد فقط من تطابقه مع ما يلي:

```
Private Sub dirDirectory_Change()
    filFiles.Path = dirDirectory.Path
    LblDirName.Caption = dirDirectory.Path
End Sub
```

❑ عدّل الإجراء cboFileType_Click() بحيث يبدو كما يلي:

```
Private Sub cboFileType_Click()
    Dim PatternPos1 As Integer
    Dim PatternPos2 As Integer
    Dim PatternLen  As Integer
    Dim Pattern     As String

    PatternPos1 = InStr(1, cboFileType.Text, "(") + 1
    PatternPos2 = InStr(1, cboFileType.Text, ")") - 1

    PatternLen = PatternPos2 - PatternPos1 + 1
    Pattern = Mid(cboFileType.Text, PatternPos1, PatternLen)

    filFiles.Pattern = Pattern
End Sub
```

❑سنحافظ على نص الإجراء filFiles_Click() بدون تغير، تأكد فقط من تطابقه مع ما يلي:

```
Private Sub filFiles_Click()
    txtFileName.Text = filFiles.filename
End Sub
```

❑عدّل نص الإجراء cmdOk_Click() للنموذج frmGetFile بحيث يصبح كما يلي:

```
Private Sub cmdOk_Click()
    Dim PathAndName As String
    Dim Path As String
    '
    If txtFileName.Text = "" Then
      MsgBox "يجب أن تختار ملفاً!", _
              vbMsgBoxRight Or vbMsgBoxRtlReading, _
             "برنامج الحجم"
      Exit Sub
    End If

    If Right(filFiles.Path, 1) <> "\" Then
      Path = filFiles.Path + "\"
    Else
```

```
    Path = filFiles.Path
  End If

  If txtFileName.Text =  filFiles.FileName Then
    PathAndName = Path + filFiles.FileName
  Else
    PathAndName = txtFileName.Text
  End If

  frmGetFile.Tag= PathAndName
  frmGetFile.Hide
End Sub
```

❑ سنحافظ على نص filFiles_DblClick() بدون تغيير، لهذا تأكد من مطابقته لما يلي:

```
Private Sub filFiles_DblClick()
  txtFileName.Text = filFiles.filename
  cmdOk_Click
End Sub
```

❑ عدّل الإجراء cmdCancel_Click() للنموذج frmGetFile بحيث يصبح كما يلي:

```
Private Sub cmdCancel_Click()
  frmGetFile.Tag = ""
  frmGetFile.Hide
End Sub
```

❑ احفظ المشروع بالطريقة المعتادة.

تنفيذ برنامج اختيار ملف

❑ اختر العنصر **Select File** من القائمة **File**.

*يظهر مربع الحوار المدعو **اختيار ملف** على الشاشة بصفة مشروطة Modal، وبهذا طالما أن مربع الحوار مفتوح، لا يمكنك العودة إلى البرنامج. وسوف تسمع صوت رنين Beep إذا نقرت بواسطة الفأرة مثلاً على نافذة البرنامج.*

تمرن على شتى عناصر التحكم الموجودة في مربع الحوار **اختيار ملف**.

- أنهِ برنامج اختيار ملف، باختيار **خروج** من القائمة **ملف**.

كيف يعمل برنامج اختيار ملف

يستخدم برنامج اختيار ملف، النموذج frmGetFile كمربع حوار Modal، عندما يحتاج إلى تزويده باسم ملف.

إجراءات النموذج frmSelect

تشرح الفقرتان التاليتان كيفية عمل الإجراءين Form_Load() و mnuSelectFile_Click() للنموذج frmSelect.

نص الإجراء Form_Load()

يُنفَذ هذا الإجراء عند بدء تشغيل البرنامج، ويتم بواسطته تحميل مربع الحوار frmGetFile وتجهيز مربع أنواع الملفات cboFileType التابع له:

```
Private Sub Form_Load()
    Load frmGetFile
    frmGetFile.cboFileType.AddItem "All files(*.*)"
    frmGetFile.cboFileType.AddItem "Text files(*.TXT)"
    frmGetFile.cboFileType.AddItem "Doc files(*.DOC)"
    frmGetFile.cboFileType.ListIndex = 0
End Sub
```

تُستخدم عبارة Load لتحميل مربع الحوار frmGetFile إلى الذاكرة.

لا يؤدي تحميل النموذج في الذاكرة، إلى إظهاره على الشاشة، وإنما يحمّل إلى الذاكرة بغية تمكين الإجراءات الأخرى من إظهاره بدون تأخير، وإمكانية التعامل مع عناصره أيضاً.

يجُهّز الإجراء بعد فراغه من تحميل مربع الحوار frmGetFile، مربع أنواع الملفات cboFileType للنموذج frmGetFile، فيملؤه بثلاثة بنود:

- All Files (*.*).
- Text File (*.TXT).

- DOC Files (*.Doc).

ثم يتم إسناد القيمة صفر إلى الخاصية ListIndex للمربع cboFileType، للدلالة على إظهار أول بند من هذه البنود وهو (*.*) All Files.

نص الإجراء mnuSelectFile _Click()

يُنفذ هذا الإجراء عند اختيار البند **اختيار ملف** من القائمة **ملف**، يُظهر هذا الإجراء النموذج frmGetFile كمربع حوار محدد أو مشروط (Modal)، ثم يستخدم مربع الحوار لإيجاد الملف الذي اختاره المستخدم من خَرْج مربع الحوار. يُقدَم خَرْج مربع الحوار (اسم الملف الذي اختاره المستخدم) عبر الخاصية Tag للنموذج frmGetFile:

```
Private Sub mnuSelectFile_Click()
    frmGetFile.Caption = "اختيار ملف"
    frmGetFile.Show 1
    If frmGetFile.Tag = "" Then
      MsgBox "لم تختر أي ملف"
    Else
      MsgBox "لقد اخترت الملف" + frmGetFile.Tag
    End If
End Sub
```

تُسنِد العبارة الأولى في الإجراء، العنوان **اختيار ملف** إلى الخاصية Caption للنموذج frmGetFile:

```
    frmGetFile.Caption = "إختيار ملف"
```

ثم يلي ذلك إظهار النموذج على الشاشة كمربع حوار محدد أو مشروط (Modal)، وذلك باستخدام الطريقة Show:

```
    frmGetFile.Show 1
```

بما أن القيمة 1 تمثل قيمة الوسيط Style للطريقة Show، فإن هذا يعنى يعني أن مربع الحوار، سوف يظهر كنموذج مشروط (Modal).

يُسند نص برنامج النموذج frmGetFile اسم الملف المنتقى، إلى الخاصية Tag لهذا النموذج.

وعند عدم اختيار ملف (أو نقر الزر **إلغاء** مثلاً)، فإن الخاصية Tag تُسنَد لها سلسلة خالية (Null).

لذلك تُفحص قيمة الخاصية Tag، فإذا احتوت على سلسلة نصية، كان ذلك هو اسم الملف المنتقى، ويتم عرضه في رسالة للمستخدم، أما إذا احتوت على سلسلة خالية Null، فهذا يعني أن المستخدم لم يختر ملفاً أو أنه نقر الزر **إلغاء**، وفي كلتا الحالتين، تُعرض رسالة على المستخدم، تفيد بأنه لم يختر ملفاً:

```
If frmGetFile.Tag = "" Then
  MsgBox "لم تختر أي ملف"
Else
  MsgBox "لقد اخترت الملف" + frmGetFile.Tag
End If
```

إجراءات النموذج frmGetFile

توضح الفقرات القادمة عمل نصوص إجراءات النموذج frmGetFile.

نص الإجراء ()Form_Load

يُنفّذ الإجراء ()Form_Load عند تحميل النموذج frmGetFile. يجدّد هذا الإجراء الخاصية Caption للافتة lblDirName، بتكليفها قيمة المسار Path لمربع سرد الأدلة:

```
Private Sub Form_Load()
    lblDirName.Caption = dirDirectory.Path
End Sub
```

نص الإجراء ()drvDrive_Change

نص هذا الإجراء هو نفس نص الإجراء ()drvDrive_Change في برنامج الحجم.

نص الإجراء ()dirDirectory_Change

نص هذا الإجراء مطابق لنص الإجراء ()dirDirectory_Change في برنامج الحجم.

نص الإجراء ()cboFileType_Click

يُنفّذ الإجراء ()cboFileType_Click عند إجراء اختيار من المربع cboFileType، ويُجدّد نص هذا الإجراء، الخاصية Pattern لمربع سرد الملفات:

```
Private Sub cboFileType_Click()
    Dim PatternPos1 As Integer
    Dim PatternPos2 As Integer
    Dim PatternLen  As Integer
    Dim Pattern     As String

    PatternPos1 = InStr(1, cboFileType.Text, "(") + 1
    PatternPos2 = InStr(1, cboFileType.Text, ")") - 1

    PatternLen = PatternPos2 - PatternPos1 + 1
    Pattern = Mid(cboFileType.Text, PatternPos1, PatternLen)

    filFiles.Pattern = Pattern
End Sub
```

تحتوي الخاصية Text لمربع سرد أنواع الملفات cboFileType (cboFileType.Text) على نوع الملف الذي اخترته.

فمثلاً، إذا كانت cboFileType.Text مساوية إلى (Text Files (*.TXT، عندها تكون الخاصية Pattern تساوي *.TXT.

يعمل الإجراء على تحديد موقع أول حرف من أحرف العينة (التي تساوي *.TXT)، وذلك بتحديد موقع رمز أول قوس هلالى افتتاحى يصادفه، وإضافة واحد إليه:

```
    PatternPos1 = InStr(1, cboFileType.Text, "(") + 1
```

وبهذه الطريقة يتمكن من معرفة موقع الرمز (*).

بشكل مماثل، يحدد الإجراء موقع آخر حرف من أحرف العينة (التي تساوي *.TXT)، وذلك بتحديد موقع رمز أول قوس هلالي إغلاقي يصادفه، وإنقاص واحد منه:

```
    PatternPos2 = InStr(1, cboFileType.Text, ")") - 1
```

ثم يحسب طول العينة، بطرح قيمة المتحول PatternPos1 من قيمة المتحول PatternPos2، ثم يجمع الناتج مع الواحد أي:

```
    PatternLen = PatternPos2 - PatternPos1 + 1
```

وأخيراً، يستخلص نص العينة، باستخدام التابع الوظيفي ()Mid:

```
Pattern = Mid(cboFileType.Text, PatternPos1, PatternLen)
```

تُسنِد العبارة الأخيرة في الإجراء، العينة المستخلصة، إلى الخاصية Pattern لمربع سرد الملفات.

```
filFiles.Pattern = Pattern
```

ونتيجة الأمر، يُظهر مربع سرد الملفات، الملفات التي تطابق العينة المستخلصة فقط.

نص الإجراء ()filFiles_Click

نص هذا الإجراء هو نفس نص الإجراء ()filFiles_Click في برنامج الحجم.

نص الإجراء ()cmdOk_Click

ينفذ هذا الإجراء عند النقر بالفأرة على الزر **موافق**:

```
Private Sub cmdOk_Click()
    Dim PathAndName As String
    Dim Path As String
    If txtFileName.Text = "" Then
      MsgBox "يجب أن تختار ملفاً!", _
             vbMsgBoxRight Or vbMsgBoxRtlReading, _
             "برنامج الحجم"
      Exit Sub
    End If
    If Right(filFiles.Path, 1) <> "\" Then
      Path = filFiles.Path + "\"
    Else
      Path = filFiles.Path
    End If
    If txtFileName.Text =  filFiles.FileName Then
      PathAndName = Path + filFiles.FileName
    Else
      PathAndName = txtFileName.Text
    End If

    frmGetFile.Tag= PathAndName
    frmGetFile.Hide
```

```
End Sub
```

كما تلاحظ، هذا الإجراء مشابه كثيراً للإجراء cmdOk_Click() في برنامج الحجم، وأول شيء يفعله الإجراء هو التأكد من اختيار ملف، وذلك بمقارنة الخاصية Text لمربع النص txtFileName مع السلسلة الخالية Null. فإذا حدث تطابق، فهذا يعني أن المستخدم لم ينتق شيئاً وتظهر رسالة خطأ وينتهي الإجراء:

```
If txtFileName.Text = "" Then
  MsgBox "!يجب أن تختار ملفاً", _
         vbMsgBoxRight Or vbMsgBoxRtlReading, _
         "برنامج الحجم"
  Exit Sub
End If
```

وبعد التحقق من اختيار ملف، يُحدّث المتحول Path، بإسناد مسار الملف المنتقى إليه (اسم الدليل). يُستخدم التابع الوظيفي Right() للتحقق من أن آخر رمز في مسار الملف المنتقى، هو الرمز (\)، فإذا لم يكن كذلك، فلا بد من إضافته إلى نهاية المسار:

```
If Right(filFiles.Path, 1) <> "\" Then
  Path = filFiles.Path + "\"
Else
  Path = filFiles.Path
End If
```

تعتبر العبارة السابقة لازمة، لأن الخاصية filFiles.Path تحتوي على الرمز (\) إضافة لحرف محرك الأقراص، وذلك عند اختيار ملف من الدليل الرئيسي لمحرك الأقراص (مثلاً \:C).

أما عند اختيار الملف من مكان آخر غير الدليل الرئيسي، فلن تحتوي الخاصية filFiles.Path على الرمز (\) في نهايته (ستكون مثلاً C:\TRY بدلاً من C:\TRY\)، ولهذا لا بد من استخدام عبارة الشرط If لضمان وجود الرمز (\) بغض النظر عن مكان اختيار الدليل.

يُصبح بالإمكان تحديث المتحول PathAndName، حالما يصبح المتحول Path جاهزاً. وحسبما يتضح من اسم المتحول PathAndName فإنه يفترض أن يحوي على الاسم

الكامل للملف (أي المسار إضافة إلى اسم الملف). تستخدم عبارة If لإسناد الاسم الكامل للملف إلى المتحول PathAndName:

```
If txtFileName.Text =  filFiles.FileName Then
  PathAndName = Path + filFiles.FileName
Else
  PathAndName = txtFileName.Text
End If
```

تتحقق عبارة If هذه بأن الملف المضاء حالياً في مربع سرد الملفات، هو نفسه الملف الوارد في مربع النص **اسم الملف**. فإذا لم يكن التطابق موجوداً فهذا يعني أن المستخدم أدخل المسار واسم الملف يدوياً، ولهذا يتم تحديث المتحول PathAndName، وإسناد السلسلة التي أدخلها المستخدم إليه.

أما عند حدوث التطابق، فستسند السلسلة Path + filFiles.FileName إلى المتحول PathAndName.

تُستخدم قيمة المتحول PathAndName بعد ذلك، لإسنادها إلى الخاصية Tag للنموذج frmGetFile:

```
frmGetFile.Tag = PathAndName
```

تُستخدم الخاصية Tag للنموذج لحفظ خرْج النموذج frmGetFile.

يعلم الإجراء الذي أظهر النموذج frmGetFile، بواسطة هذه الخاصية frmGetFile.Tag، اسم ومسار الملف الذي اختاره المستخدم.

تُزيل آخر عبارة في الإجراء، النموذج من الشاشة، بواسطة الطريقة Hide:

```
frmGetFile.Hide
```

تقوم الطريقة Hide بإخفاء النموذج فقط من الشاشة، دون إزالته من الذاكرة.

وتعود السيطرة إلى الإجراء الذي أظهر النموذج frmGetFile، لإكمال تنفيذ باقي تعليماته.

نص الإجراء ()filFiles_DblClick

نص هذا الإجراء يطابق تماماً ذاك الموجود في برنامج الحجم.

نص الإجراء ()cmdCancel_Click

يُنفذ الإجراء ()cmdCancel_Click عند نقر الزر **إلغاء:**

```
Private Sub cmdCancel_Click()
    frmGetFile.Tag = ""
    frmGetFile.Hide
End Sub
```

اتفقنا أن الخاصية Tag تُستخدم للاحتفاظ بمسار واسم الملف المنتقى. تُسند سلسلة خالية Null إلى هذه الخاصية، وذلك لأن المستخدم نقر الزر **إلغاء،** بمعنى أنه أراد الخروج دون اختيار ملف معين.

```
    frmGetFile.Tag = ""
```

تُزيل آخر عبارة في الإجراء النموذج frmGetFile من الشاشة باستخدام الطريقة Hide:

```
    frmGetFile.Hide
```

وبعد تنفيذ هذه العبارة، يُزال النموذج frmGetFile من الشاشة، وتعود السيطرة ثانية إلى الإجراء الذي أظهر النموذج frmGetFile.

الخلاصة

تعلمنا من هذا الفصل، كيفية إستخدام عناصر تحكم نظام الملفات، لكتابة برامج تمكنك من إختيار الملفات، كما تعلمنا أيضاً كيفية كتابة نموذج عام الغرض (النموذج frmGetfile) لإستخدامه من قبل أى برنامج يتطلب إختيار ملفات.

الفصل الرابع عشر

الوصول إلى الملفات

تحتاج الكثير من البرامج إلى قراءة وكتابة البيانات في الملفات على الأقراص. سنتعلم في هذا الفصل كيفية تشكيل الملفات وكيفية قراءة البيانات من الملفات، وكتابة البيانات عليها.

هنالك ثلاث طرق للوصول إلى الملفات:

- الوصول العشوائي Random Access.
- الوصول التسلسلي Sequential Access.
- الوصول الثنائي Binary Access.

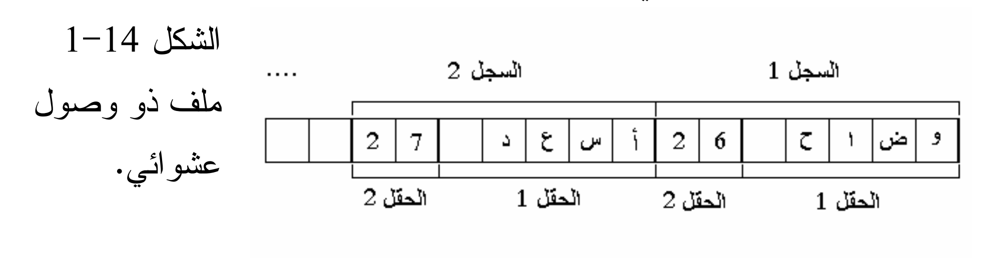
يعلمك هذا الفصل كيف تستخدم كل تقنية من تقنيات الوصول إلى الملفات، حتى تستطيع التعامل معها.

الملفات ذات الوصول العشوائي

يشبه هذا الملف قاعدة البيانات Database، فهو مؤلف من سجلات متطابقة الحجم، وكل سجل مؤلف من حقول تستخدم لحفظ البيانات، يبين الشكل 14-1 ملف ذي وصول عشوائي، يتألف كل سجل من سجلاته من حقلين، الحقل الأول عبارة عن سلسلة بطول 5 بايتات مخصصة لحفظ اسم الشخص، والحقل الثاني عبارة عن سلسلة بطول بايتين، يحفظ فيها عمر الشخص.

إذاً يبلغ طول كل سجل سبعة بايتات، تشكّل البايتات السبعة الأولى، أول سجل. والبايتات السبعة الثانية، ثاني سجل. وهكذا. ويحفظ كل سجل بيانات شخص ما.

الشكل 14-1
ملف ذو وصول عشوائي.



برنامج الهاتف

يوضح برنامج الهاتف، كيفية إنشاء ومعالجة الملفات ذات الوصول العشوائي. يمكنك البرنامج من التعامل مع ملف قاعدة بيانات يدعى Phone.DAT، يحتفظ بسجلات الأشخاص وأرقام هواتفهم.

التمثيل المرئي لبرنامج الهاتف

- أنشئ مشروعاً جديداً من النوع Standard EXE.
- أنشئ الدليل C:\VB5Prg\Ch14. واحفظ نموذج المشروع بالاسم Phone.Frm في الدليل C:\VB5Prg\Ch14، واحفظ ملف المشروع باسم Phone.Vbp في نفس الدليل.
- أنشئ النموذج طبقاً للجدول 14-1.

يُفترض أن يظهر النموذج المكتمل كما في الشكل 14-2.

الجدول 14-1. جدول خصائص برنامج الهاتف.

الكائن	الخاصية	القيمة
Form	**Name**	**frmPhone**
	Caption	(اجعله فارغاً)
	RightToLeft	True
CommandButton	**Name**	**cmdNew**

	Caption	&جديد
	RightToLeft	True
CommandButton	**Name**	**cmdNext**
	Caption	ال&تالي
	RightToLeft	True
CommandButton	**Name**	**cmdPrevious**
	Caption	ال&سابق
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
TextBox	**Name**	**txtName**
	Text	(اجعله فارغاً)
	MaxLength	40
	RightToLeft	True
TextBox	**Name**	**txtPhone**
	Text	(اجعله فارغاً)
	MaxLength	100
	RightToLeft	True
TextBox	**Name**	**txtComments**
	Text	(اجعله فارغاً)
	MaxLength	40

الكائن	الخاصية	القيمة
	MultiLine	True
	ScrollBars	2-Vertical
	RightToLeft	True
Label	**Name**	**lblName**
	Caption	الاسم:

	RightToLeft	True
Label	**Name**	**lblPhone**
	Caption	رقم الهاتف:
	RightToLeft	True
Label	**Name**	**lblComments**
	Caption	ملاحظات:
	RightToLeft	True

إدخال نص برنامج الهاتف

سنحتاج مع النموذج PHONE.FRM إلى وحدة نمطية Module. وسنوضح سبب الحاجة إليه لاحقاً، أما الآن فإليك طريقة بنائه:

- اختر **Add Module** من القائمة **Project**.

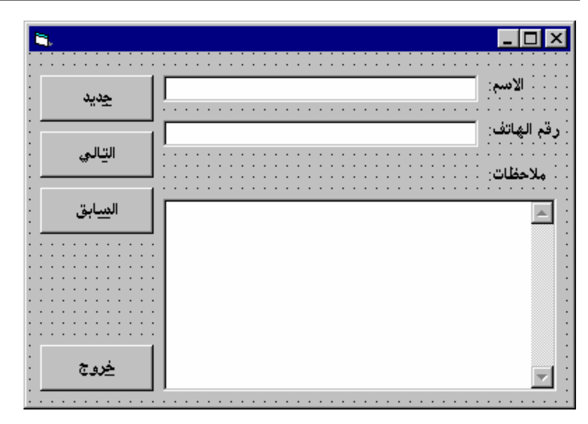
يستجيب فيجول بيسك بإظهار مربع الحوار Add Module.

- اختر الرمز Module من الصفحة New لمربع الحوار Add Module، ثم انقر الزر **فتح**.

يستجيب فيجول بيسك بإنشاء وحدة نمطية جديدة، ويُظهر إطار نص البرنامج الخاص بها، وكما تلاحظ يطلق فيجول بيسك عليه التسمية Module1.

- احفظ الوحدة النمطية الجديدة بالاسم PHONE.BAS كالتالي:
- اختر **Save Module1 As** من القائمة **File** واحفظ الملف بالاسم Phone.BAS في الدليل C:\VB5Prg\Ch14.
- أدخل النص التالي في قسم التصاريح العامة للوحدة نمطية Phone.BAS:

```
'يجب التصريح عن كل المتحولات
Option Explicit
Type PersonInfo
  Name      As String * 40
  Phone     As String * 40
  comments  As String * 100
End Type
```



الشكل 14-2

نموذج برنامج الهاتف.

هذا هو كل النص الذي نحتاج كتابته في الوحدة النمطية. سنباشر الآن كتابة نص النموذج frmPhone:

- أدخل النص التالي في قسم التصاريح العامة للنموذج frmPhone:

```
'يجب التصريح عن كل المتحولات
Option Explicit
Dim gPerson As PersonInfo
Dim gFileNum As Integer
Dim gRecordLen As Long
Dim gCurrentRecord As Long
Dim gLastRecord As Long
```

- أنشئ إجراءً جديداً في النموذج frmPhone (باختيار **Add Procedure** من القائمة **Tools**) وسمّه SaveCurrentRecord().

- أدخل النص التالي في الإجراء SaveCurrentRecord() الذي أنشأناه للتو:

```
Public Sub SaveCurrentRecord()
    gPerson.Name = txtName.Text
    gPerson.Phone = txtPhone.Text
    gPerson.comments = txtComments.Text
    Put #gFileNum, gCurrentRecord, gPerson
End Sub
```

- أنشئ إجراءً جديداً في النموذج frmPhone وسمّه ShowCurrentRecord().

- أدخل النص التالي في الإجراء ShowCurrentRecord() الذي أنشأناه للتو:

```
Public Sub ShowCurrentRecord()
    Get #gFileNum, gCurrentRecord, gPerson
    txtName.Text = Trim(gPerson.Name)
    txtPhone.Text = Trim(gPerson.Phone)
```

```
    txtComments.Text = Trim(gPerson.comments)
    frmPhone.Caption = "السجل:" + Str(gCurrentRecord) + _
                       "/" + Str(gLastRecord)
End Sub
```

- أدخل النص التالي في الإجراء ()Form_Load:

```
Private Sub Form_Load()
    gRecordLen = Len(gPerson)
    gFileNum = FreeFile
    Open "PHONE.DAT" For Random As gFileNum Len = gRecordLen
    gCurrentRecord = 1
    gLastRecord = FileLen("PHONE.DAT") / gRecordLen
    If gLastRecord = 0 Then
      gLastRecord = 1
    End If
    ShowCurrentRecord
End Sub
```

- أدخل النص التالي ضمن الإجراء ()cmdNew_Click:

```
Private Sub cmdNew_Click()
    SaveCurrentRecord
    gLastRecord = gLastRecord + 1
    gPerson.Name = ""
    gPerson.Phone = ""
    gPerson.comments = ""
    Put #gFileNum, gLastRecord, gPerson
    gCurrentRecord = gLastRecord
    ShowCurrentRecord
    txtName.SetFocus
End Sub
```

- أدخل النص التالي ضمن الإجراء ()cmdNext_Click:

```
Private Sub cmdNext_Click()
    If gCurrentRecord = gLastRecord Then
      Beep
      MsgBox "end of File!", vbExclamation
    Else
      SaveCurrentRecord
      gCurrentRecord = gCurrentRecord + 1
```

```
      ShowCurrentRecord
    End If
    txtName.SetFocus
End Sub
```

❑ أدخل النص التالي ضمن الإجراء cmdPrevious_Click():

```
Private Sub cmdPrevious_Click()
    If gCurrentRecord = 1 Then
      Beep
      MsgBox "Beginning of file!", vbExclamation
    Else
      SaveCurrentRecord
      gCurrentRecord = gCurrentRecord - 1
      ShowCurrentRecord
    End If
    txtName.SetFocus
End Sub
```

❑ أدخل النص التالي ضمن الإجراء cmdExit_Click():

```
Private Sub cmdExit_Click()
    SaveCurrentRecord
    Close #gFileNum
    End
End Sub
```

تنفيذ برنامج الهاتف

لنشاهد نتيجة ما كتبناه:

❑ نفّذ برنامج الهاتف.

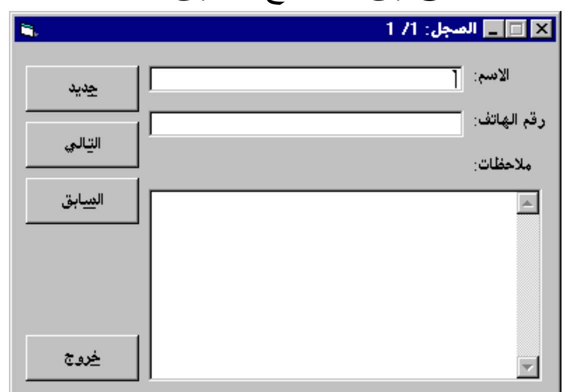
❑ لاحظ المظاهر التالية أثناء تشغيل برنامج الهاتف.

عند تنفيذ البرنامج للمرة الأولى، لا تكون هناك بيانات، ويظهر سجل فارغ على الشاشة، وهو السجل1/1، حسب ما يشار إليه بواسطة عنوان إطار البرنامج. يمكّنك البرنامج من إدخال اسم الشخص ورقم هاتفه وملاحظات حوله، (انظر الشكل 14-3) تستطع الانتقال من حقل لآخر باستخدام الفأرة أو بواسطة المفتاح Tab.

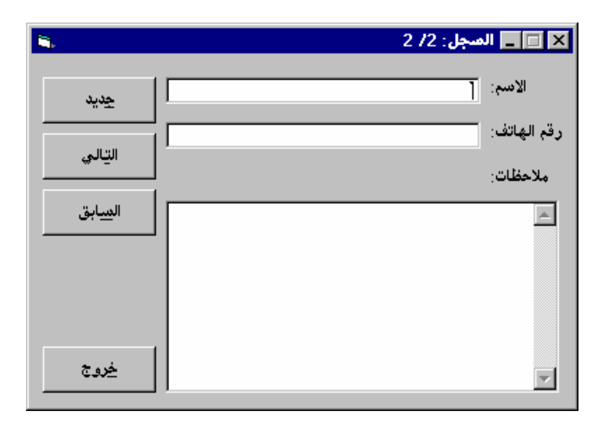
❑ انقر الزر **جديد** لإضافة سجل جديد.

يظهر سجل فارغ. ويُظهر عنوان البرنامج، رقم السجل الجديد وعدد السجلات الإجمالي (انظر الشكل14-4).

حالما يصبح لديك عدة سجلات تستطيع الانتقال من سجل لآخر، باستخدام زري الأمر **التالي و السابق**. فنقر الزر **التالي** ينقلك إلى السجل التالي، وعلى العكس يؤدي نقر الزر **السابق** إلى الرجوع بك إلى السجل السابق.



الشكل 14-3
السجل الأول في برنامج الهاتف.



الشكل 14-4
برنامج الهاتف بعد إضافة سجل جديد إليه.

- انقر الزر **خروج** لإنهاء برنامج الهاتف.

تحفظ كافة السجلات التي أضفتها في الملف Phone.DAT وسوف تشاهد هذه السجلات عندما نشغل برنامج الهاتف في المرة القادمة.

كما تلاحظ لا يمكنك حذف أو البحث عن سجل، في برنامج الهاتف هذا، سنحسّن هذا البرنامج لاحقاً، لتمكينك من حذف سجلات أو البحث عن سجل محدد.

كيف يعمل برنامج الهاتف

يفتح برنامج الهاتف الملف Phone.DAT كملف عشوائي ذي ثلاثة حقول (الاسم، رقم الهاتف، ملاحظات)، وإذا كان الملف غير موجود فإنه يقوم بإنشائه آلياً.

التصريح عن تركيب معرّف من قبل المستخدم

يصرح النص الذي كتبناه في قسم التصاريح العامة للوحدة النمطية، عن تركيب Structure معرّف من قبل المستخدم، هذا النوع من المتحولات هو نوع جديد، ينطوي تحته تعريف ثلاثة حقول، تشكل بدورها سجلاً في الملف Phone.DAT:

```
Type PersonInfo
  Name      As String * 40
  Phone     As String * 40
  comments  As String * 100
End Type
```

اسم التركيب المصرّح عنه هو PersonInfo، ويتألف من ثلاثة متحولات:

- Name الاسم: عبارة عن سلسلة نصية بطول 40 حرفاً.
- Phone الهاتف: سلسلة نصية بطول 40 حرفاً.
- Comments التعليقات: سلسلة بطول 100 حرف.

سيستخدم برنامج الهاتف لاحقاً، متحولاً من نوع التركيب PersonInfo لكتابة وقراءة البيانات على الملف Phone.DAT.

لاحظ أن التصريح عن التركيب PersonInfo تم في قسم التصاريح العامة للوحدة النمطية، وليس في قسم التصاريح العامة للنموذج frmPhone.

قسم التصاريح العامة للنموذج frmPhone

هنالك مجموعة من التصاريح عن متحولات، يفترض أنها ستصبح مرئية لكل إجراءات هذا النموذج:

```
'يجب التصريح عن كل المتحولات
Option Explicit
Dim gPerson As PersonInfo
Dim gFileNum As Integer
```

```
    Dim gRecordLen As Long
    Dim gCurrentRecord As Long
    Dim gLastRecord As Long
```

يُصرَح عن المتحول gPerson كمتحول من نوع التركيب PersonInfo، وباعتبار أن التركيب PersonInfo يتألف من ثلاثة حقول هي Name، Phone، Comments. فإن المتحول gPerson يتألف بدوره من ثلاثة متحولات هي gPerson.Name، Person.Phone، gPerson.Comments، حيث يستخدم المتحول gPerson للاحتفاظ ببيانات السجل.

فتح الملف Phone.DAT

لا بد بالطبع من فتح الملف قبل القراءة منه أو الكتابة عليه، وجزء البرنامج المسئول عن فتح الملف Phone.DAT يقع ضمن الإجراء Form_Load():

```
Private Sub Form_Load()
    gRecordLen = Len(gPerson)
    gFileNum = FreeFile
    Open "PHONE.DAT" For Random As gFileNum Len = gRecordLen
    gCurrentRecord = 1
    gLastRecord = FileLen("PHONE.DAT") / gRecordLen
    If gLastRecord = 0 Then
      gLastRecord = 1
    End If
    ShowCurrentRecord
End Sub
```

نحتاج إلى معرفة طول كل سجل من سجلات الملف، ورقم الملف، لنتمكن من فتح ملف للوصول العشوائي، وهكذا وقبل فتح الملف، يستخلِص الإجراء هاتين القيمتين بواسطة العبارتين التاليتين:

```
    gRecordLen = Len(gPerson)
    gFileNum = FreeFile
```

تستخدِم العبارة الأولى التابع الوظيفي Len()، لمعرفة طول المتحول gPerson، وباعتبار أن المتحول gPerson يمثل سجلاً في الملف Phone.DAT، فإن طوله هو نفس طول السجل.

تستخدم العبارة الثانية التابع الوظيفي FreeFile للحصول على رقم ملف حر غير مستخدم، وحسب ما أوضحناه سابقاً، نحتاج عند فتح ملف ما، إلى تحديد رقم تعريف الملف المفتوح. لأن العبارات اللاحقة في البرنامج، والتي ستنجز عمليات معينة على الملف، تحتاج إلى رقم الملف هذا، لإخبار فيجول بيسك بالملف الذي ستنجز العملية عليه. في الحقيقة يعتبر هذا الرقم كمقبض للملف المفتوح.

والآن، تمت معرفة قيمة المتحولين gRecordLen و gFileNum، وأصبح فتح الملف Phone.DAT للوصول العشوائي جاهزاً، وذلك بواسطة العبارة التالية:

```
Open "POHNE.DAT" For Random As gFileNum Len = gRecordLen
```

تُنشئ عبارة Open الملف إذا لم يكن موجوداً أصلاً. وهكذا وعند أول تشغيل لبرنامج الهاتف، يتم إنشاء الملف Phone.DAT في نفس الدليل الذي يُنفذ فيه برنامج الهاتف.

سنحول برنامج الهاتف إلى ملف تنفيذي Phone.EXE حال انتهائنا من تطويره، فإذا حفظت البرنامج Phone.EXE على سبيل المثال في الدليل C:\TRY، ثم نفذّته انطلاقاً من هذا الدليل، فسوف تلاحظ أن الملف Phone.DAT تشكل في نفس الدليل C:\TRY.

بقي أن تعلم أن الملف Phone.DAT، سيتشكل في دليل فيجول بيسك نفسه، عندما تكون في طور التصميم (أي عندما تنفذ برنامج الهاتف ضمن بيئة فيجول بيسك).

بعد الانتهاء من فتح ملف Phone.DAT، تتم تهيئة المتحول gCurrentRecord والمتحول gLastRecord بقيم ابتدائية.

يُستخدَم المتحول gCurrentRecord لحفظ رقم السجل الحالي (السجل الظاهر حالياً)، وباعتبار أنه يتوجب إظهار السجل ذي الرقم 1 في بداية تشغيل البرنامج، لهذا فقد أُسندت القيمة الابتدائية 1 إلى المتحول gCurrentRecord.

```
gCurrentRecord = 1
```

يُستخدم المتحول gLastRecord لحفظ رقم آخر سجل في الملف (العدد الكلي للسجلات)، يُحسب عدد السجلات بقسمة طول الملف الكلي على طول السجل الواحد:

```
gLastRecord = FileLen("PHONE.DAT") / gRecordLen
```

عموماً، توجد حالة خاصة لعملية الحساب السابقة، لن تعمل. فالتابع الوظيفي FileLen() يعيد القيمة صفر، إذا كان الملف قد أنشئ للتو، ومنه فإن عملية الحساب السابقة

ستتسبب بإسناد القيمة صفر إلى المتحول gLastRecord، (نُعيد: إذا كان الملف قد أنشئ للتو).

وللتحقق من أن قيمة المتحول gLastRecord لا تساوي الصفر، تستخدم عبارة If التالية:

```
If gLastRecord = 0 Then
  gLastRecord = 1
End If
```

تتحقق عبارة If من مساواة المتحول gLastRecord للصفر، فإذا كانت هذه هي الحال، فإن عبارة If ستسند القيمة واحد إلى المتحول gLastRecord.

تُنفِذ آخر عبارة في الإجراء Form_Load() الإجراء ShowCurrentRecord:

```
ShowCurrentRecord
```

يُظهِر الإجراء ShowCurrentRecord بيانات السجل المحدد بالمتحول gCurrentRecord. وبما أن قيمة gCurrentRecord تساوي 1 فسوف يتم إظهار بيانات السجل ذي الرقم "واحد".

نص الإجراء ShowCurrentRecord()

يُظهِر هذا الإجراء بيانات السجل المحدد بالمتحول gCurrentRecord:

```
Public Sub ShowCurrentRecord()
    Get #gFileNum, gCurrentRecord, gPerson
    txtName.Text = Trim(gPerson.Name)
    txtPhone.Text = Trim(gPerson.Phone)
    txtComments.Text = Trim(gPerson.comments)
    frmPhone.Caption = "السجل:" + Str(gCurrentRecord) + "/" +_
                       Str(gLastRecord)
End Sub
```

تَستخدم أول عبارة في الإجراء، العبارة Get لملء المتحول gPerson ببيانات السجل الحالي:

```
Get #gFileNum, gCurrentRecord, gPerson
```

تأخذ العبارة Get ثلاثة وسائط، يحدد الوسيط الأول رقم الملف أو مقبضه (وهو الرقم الذي حدد عند فتح الملف)، يحدد الوسيط الثاني، رقم السجل المطلوب قراءته، ويحدد الوسيط الثالث اسم المتحول الذي ستُنسَخ إليه بيانات السجل.

فمثلاً لتكن قيمة gCurrentRecord الراهنة تساوي 5، سيحتوي المتحول gPerson بعد تنفيذ العبارة السابقة على بيانات السجل ذي الرقم 5، أي بمعنى أن الحقول gPerson.Name، gPerson.Phone، gPerson.Comments سوف تحتوي على البيانات المخزنة في الحقول: الاسم و رقم الهاتف و ملاحظات، للسجل الخامس.

وبعد ملء المتحول gPerson ببيانات السجل الحالي، يتم إظهار محتوياته، بإسناد قيم حقوله إلى مربعات النص TxrName و txtPhone و txtComments على التوالي:

```
txtName.Text = Trim(gPerson.Name)
txtPhone.Text = Trim(gPerson.Phone)
txtComments.Text = Trim(gPerson.comments)
```

نحتاج إلى بعض الشرح لتوضيح سبب استخدام Trim.

لنفترض مثلاً أن السجل الحالي يحتوي على الاسم John Smith في حقل الاسم، ستلاحظ بعد تنفيذ العبارة Get أن المتحول gPerson.Name سيتضمن الأحرف التالية:

```
"John Smith.............................."
```

وذلك بسبب أن الحقل gPerson.Name معرّف كحقل بطول 40 رمزاً، بينما يبلغ طول John Smith عشرة رموز فقط، يضيف فيجول بيسك 30 فراغ وراء الاسم في الحقل عند تخزين السجل في الملف، يجب أن لا تظهر هذه الفراغات في مربع النص عند إظهار قيمة الحقل فيه، ولهذا يُستخدم التابع الوظيفي Trim() لمنع ظهور الفراغات.

تُظهِر آخر عبارة في الإجراء، رقم السجل الحالي في شريط عنوان البرنامج:

```
frmPhone.Caption = "السجل:" + Str(gCurrentRecord) + "/" +_
                    Str(gLastRecord)
```

فمثلاً، لتكن القيمة الراهنة للمتحول gCurrentRecord تساوي 5، وقيمة gLastRecord تساوي 15، تُظهر العبارة السابقة في شريط عنوان البرنامج، العنوان "السجل:15/5".

نص الإجراء cmdNext_Click()

يُنفذ الإجراء cmdNext_Click() عند نقر الزر **التالي**. هذا الإجراء مسئول عن إظهار محتويات السجل التالي:

```
Private Sub cmdNext_Click()
    If gCurrentRecord = gLastRecord Then
      Beep
      MsgBox "end of File!", vbExclamation
    Else
      SaveCurrentRecord
      gCurrentRecord = gCurrentRecord + 1
      ShowCurrentRecord
    End If
    txtName.SetFocus
End Sub
```

يتحقق الشرط If من أن gCurrentRecord لا يشير إلى آخر سجل في الملف، فإذا كان يشير إلى نهاية الملف، يُصدر الحاسب صوت تنبيه، ويُظهر رسالة لإعلامك بأنك وصلت إلى نهاية الملف، حيث لا يوجد المزيد من السجلات:

```
Beep
MsgBox "end of File!", vbExclamation
```

أما إذا كان gCurrentRecord لا يساوي gLastRecord، فسوف تُنفذ العبارات التالية:

```
SaveCurrentRecord
gCurrentRecord = gCurrentRecord + 1
ShowCurrentRecord
```

تُنفِذ أول عبارة من هذه العبارات، الإجراء SaveCurrentRecord الذي يتولى حفظ السجل الحالي (حفظ محتويات مربعات النص txtName و txtPhone و txtComments)، ثم تزيد العبارة الثانية قيمة المتحول gCurrentRecord بمقدار 1، حتى يُشير إلى السجل التالي، وأخيراً تُنفِذ العبارة الثالثة الإجراء ShowCurrentRecord المسئول عن إظهار محتويات السجل الذي يُشير إليه المتحول gCurrentRecord حالياً.

تستخدم آخر عبارة في الإجراء طريقة SetFocus، لتوجيه تركيز لوحة المفاتيح إلى مربع النص txtName:

```
txtName.SetFocus
```

وبعد تنفيذ هذه العبارة، تظهر المشيرة في مربع النص txtName.

نص الإجراء SaveCurrentRecord

يُعّد هذا الإجراء، مسئولاً عن حفظ محتويات مربعات النص txtName و txtPhone و txtComments في السجل المحدّد بالمتحول gCurrentRecord:

```
Public Sub SaveCurrentRecord()
    gPerson.Name = txtName.Text
    gPerson.Phone = txtPhone.Text
    gPerson.Comments = txtComments.Text
    Put #gFileNum, gCurrentRecord, gPerson
End Sub
```

تملأ العبارات الثلاث الأولى، المتحول gPerson بمحتويات مربعات النص الثلاثة:

```
gPerson.Name = txtName.Text
gPerson.Phone = txtPhone.Text
gPerson.Comments = txtComments.Text
```

وبذلك تنتقل محتويات مربعات النصوص، إلى حقول المتحول gPerson.

يُنفِذ الإجراء بعد ذلك عبارة Put التي تحفظ محتويات حقول المتحول gPerson في السجل ذي الرقم gCurrentRecord:

```
Put #gFileNum, gCurrentRecord, gPerson
```

تأخذ العبارة Put ثلاثة وسائط:

- يحدد الوسيط الأول رقم الملف أو مقبضه، (الرقم الذي يتحدد عند فتح الملف).
- يحدد الوسيط الثاني رقم السجل الذي سيجري حفظه.
- يحدد الوسيط الثالث اسم المتحول الذي ستحفظ محتويات حقوله في السجل.

إضافة سجل جديد إلى الملف Phone.DAT

يُعّد الإجراء cmdNew_Click() مسئولاً عن إضافة سجل جديد إلى الملف Phone.DAT، ويُنفّذ بعد النقر على الزر **جديد**:

```
Private Sub cmdNew_Click()
    SaveCurrentRecord
```

```
    gLastRecord = gLastRecord + 1
    gPerson.Name = ""
    gPerson.Phone = ""
    gPerson.Comments = ""
    Put #gFileNum, gLastRecord, gPerson
    gCurrentRecord = gLastRecord
    ShowCurrentRecord
    txtName.SetFocus
End Sub
```

تُنفّذ أول عبارة في هذا الإجراء SaveCurrentRecord، بما يعني حفظ محتويات السجل الحالي (محتويات مربعات النص الحالية) في الملف Phone.DAT، وبعد حفظ السجل الحالي، يُضيف الإجراء سجلاً فارغاً جديداً إلى الملف Phone.DAT بواسطة العبارات التالية:

```
gLastRecord = gLastRecord + 1
gPerson.Name = ""
gPerson.Phone = ""
gPerson.comments = ""
Put #gFileNum, gLastRecord, gPerson
```

تزيد العبارة الأولى المتحول gLastRecord بمقدار 1، ثم تُسند القيمة لاشيء (سلسلة خالية)، لحقول المتحول gPerson، وأخيراً، تُستخدم عبارة Put لإنشاء السجل الجديد.

يتحدد رقم السجل الجديد بالمتحول gLastRecord، كما أن محتوياته هي محتويات حقول المتحول gPerson (أي السلاسل الفارغة Null).

وبعد إنشاء السجل الفارغ الجديد، يتم تحديث المتحول gCurrentRecord بحيث يشير إلى السجل الجديد:

```
gCurrentRecord = gLastRecord
```

ثم يُنفذ الإجراء ShowCurrentRecord لإظهار السجل الذي أنشئ للتو، وأخيراً يستخدم الإجراء العبارة:

```
txtName.SetFocus
```

لنقل تركيز لوحة المفاتيح إلى مربع النص txtName، فتظهر المشيرة فيه، بعد تنفيذ هذه العبارة.

نص الإجراء cmdPrevious_Click()

يُنفَذ هذا الإجراء عند نقر الزر **السابق**، يُعّد هذا الإجراء مسئولاً عن إظهار محتويات السجل السابق:

```
Private Sub cmdPrevious_Click()
    If gCurrentRecord = 1 Then
      Beep
      MsgBox "Beginning of file!", vbExclamation
    Else
      SaveCurrentRecord
      gCurrentRecord = gCurrentRecord - 1
      ShowCurrentRecord
    End If
    txtName.SetFocus
End Sub
```

يتحقق الإجراء أولاً، من أن السجل الحالي ليس السجل الأول، فإذا كان السجل الأول، فلا يمكن الرجوع وراء ذلك، لهذا يُصدر الحاسب صوتاً تنبيهاً، ويُظهر رسالة تُعلم المستخدم بأن السجل الحالي يمثل بداية الملف:

```
Beep
MsgBox "Beginning of file!", vbExclamation
```

بينما تُنفّذ العبارات التالية إذا لم يكن السجل الحالي هو السجل الأول:

```
SaveCurrentRecord
gCurrentRecord = gCurrentRecord - 1
ShowCurrentRecord
```

تُنفّذ العبارة الأولى، الإجراء SaveCurrentRecord، لحفظ السجل الحالي في الملف Phone.DAT، وتنقص العبارة الثانية قيمة المتحول gCurrentRecord بمقدار 1، ثم ينفذ الإجراء ShowCurrentRecord لإظهار محتويات السجل السابق في مربعات النص: txtName و txtPhone و txtComments.

وأخيراً، تُسلط آخر عبارة في الإجراء، تركيز لوحة المفاتيح على مربع النص txtName:

```
txtName.SetFocus
```

لتظهر المشيرة في مربع النص txtName بعد تنفيذ هذه العبارة.

نص الإجراء ()cmdExit_Click

ينفذ الإجراء ()cmdExit_Click عند نقر الزر **خروج**، يحفظ هذا الإجراء أولاً، السجل الراهن في الملف Phone.DAT ثم يغلق هذا الملف، وينهي البرنامج:

```
Private Sub cmdExit_Click()
    SaveCurrentRecord
    Close #gFileNum
    End
End Sub
```

تُغلق عبارة Close الملف Phone.DAT، وكما تُلاحظ فإنها تأخذ وسيطاً واحداً يحدد رقم الملف المطلوب إغلاقه.

لست بحاجة حقيقة إلى استخدام عبارة Close في هذا المثال، لأن عبارة End تُغلق كافة الملفات بما فيها الملفات المفتوحة بواسطة عبارة Open، كما أنها تنهي البرنامج، ولكن وضعنا العبارة Close في هذا المكان، لتوضيح كيفية إغلاق ملف، دون استخدام العبارة End (بدون إنهاء البرنامج ككل).

تحسين برنامج الهاتف

سنحسن البرنامج الآن بإضافة زري الأمر **بحث** و**حذف**، يمكّنك الزر **بحث**، من البحث عن اسم محدد ويمكّنك الزر **حذف** من حذف سجل معين:

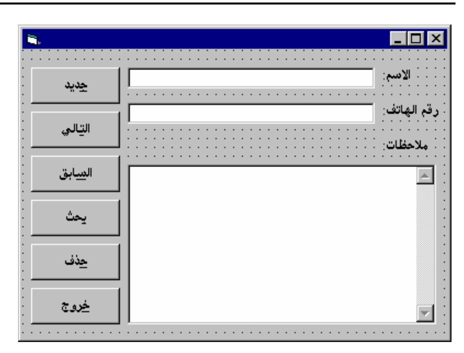
- أضف زر أمر إلى النموذج frmPhone، وأسند له الخصائص التالية:

```
Name:           cmdSearch
Caption:        &بحث
RightToLeft:    True
```

- أضف زر أمر أخر إلى النموذج frmPhone وأسند له الخصائص التالية:

```
Name:           cmdDelete
Caption:        &حذف
RightToLeft:    True
```

يفترض أن يبدو النموذج الآن كما في الشكل14-5:



الشكل 14-5
النموذج frmPhone بعد إضافة
الزرين **بحث** و **حذف** إليه.

❑ أدخل النص التالي ضمن الإجراء ()cmdSearch_Ckick:

```
Private Sub cmdSearch_Click()
    Dim NameToSearch As String
    Dim Found As Integer
    Dim RecNum As Long
    Dim TmpPerson As PersonInfo

    NameToSearch = InputBox("البحث عن", "بحث")
    If NameToSearch = "" Then
      txtName.SetFocus
      Exit Sub
    End If
    NameToSearch = UCase(NameToSearch)
    Found = False
    For RecNum = 1 To gLastRecord
      Get #gFileNum, RecNum, TmpPerson
      If NameToSearch=UCase(Trim(TmpPerson.Name)) Then
        Found = True
        Exit For
      End If
    Next

    If Found = True Then
      SaveCurrentRecord
      gCurrentRecord = RecNum
      ShowCurrentRecord
    Else
      MsgBox "لم أجد الاسم:" + NameToSearch
    End If
    txtName.SetFocus
```

```
End Sub
```

❑ أدخل النص التالي ضمن الإجراء cmdDelete_Ckick():

```
Private Sub cmdDelete_Click()
    Dim DirResult
    Dim TmpFileNum
    Dim TmpPerson As PersonInfo
    Dim RecNum As Long
    Dim TmpRecNum As Long
    'التأكد من عملية الحذف
    If MsgBox("هل تريد حذف السجل فعلاً؟", vbYesNo) = vbNO Then
      txtName.SetFocus
      Exit Sub
    End If
    'حذف الملف المؤقت السابق في حال وجوده
    If Dir("PHONE.TMP") = "PHONE.TMP" Then
      Kill "PHONE.TMP"
    End If
    'إنشاء الملف المؤقت
    TmpFileNum = FreeFile
    Open "PHONE.TMP" For Random As TmpFileNum Len = gRecordLen
    'نسخ محتويات ملف المعلومات إلى الملف المؤقت
    'ما عدا السجل المراد حذفه
    RecNum = 1
    TmpRecNum = 1
    Do While RecNum < gLastRecord + 1
      If RecNum <> gCurrentRecord Then
        Get #gFileNum, RecNum, TmpPerson
        Put #TmpFileNum, TmpRecNum, TmpPerson
        TmpRecNum = TmpRecNum + 1
      End If
      RecNum = RecNum + 1
    Loop
    Close gFileNum
    'حذف الملف الأصلي بعد نقل محتوياته للملف المؤقت
    Kill "PHONE.DAT"
    Close TmpFileNum
    'إعادة تسمية الملف المؤقت باسم الملف الأصلي
    Name "PHONE.TMP" As "PHONE.DAT"
```

```
    gFilleNum = FreeFile
    'إعادة فتح الملف الأصلي
    Open "PHONE.DAT" For Random As gFileNum Len = gRecordLen
    'إنقاص عدد السجلات بمقدار واحد
    gLastRecord = gLastRecord - 1

    if gLastRecord = 0 Then gLastRecord = 1
    If gCurrentRecord > gLastRecord Then
      gCurrentRecord = gLastRecord
    End If
    'إظهار السجل الحالي
    ShowCurrentRecord
    txtName.SetFocus
End Sub
```

- نفّذ برنامج الهاتف وتمرن على استخدام الزرين **بحث** و**حذف**.

نص الإجراء cmdSearch_Ckick()

يُنفذ الإجراء cmdSearch_Ckick() عند نقر الزر **بحث**، يمكّنك هذا الإجراء من البحث عن اسم محدد.

يبدأ الإجراء باستخدام التابع الوظيفي InputBox() للحصول على الاسم المراد البحث عنه، من المستخدم. يخزّن الاسم الذي أدخله المستخدم في المتحول NameToSearch:

```
NameToSearch = InputBox("البحث عن", "بحث")
```

يُعيد التابع الوظيفي InputBox() سلسلة فارغة Null في حالة النقر على زر Cancel لمربع الإدخال **بحث**، ولهذا تُستخدم عبارة If لمعرفة ما حصل، ينتهي تنفيذ الإجراء:

```
If NameToSearch = "" Then
  txtName.SetFocus
  Exit Sub
End If
```

يُستخدم التابع الوظيفي UCase() بعد ذلك، لتحويل الاسم (قيمة المتحول NameToSearch) إلى أحرف كبيرة:

```
NameToSearch = UCase(NameToSearch)
```

تعتبر عملية التحويل هذه ضرورية جداً، لأن البحث عن الاسم، يجب أن لا يكون مقيداً بحالة الحرف (كبيرة أو صغيرة)، فمثلاً، يجب العثور على السجل المحتوى على الاسم JOHN، حتى وإن أدخل المستخدم john.

ملاحظة

تعتبر عملية التحويل السابقة، غير ضرورية بالنسبة للأسماء العربية.

استخدمنا حلقة For للبحث عن الاسم الذي أدخله المستخدم:

```
Found = False
For RecNum = 1 To gLastRecord
  Get #gFileNum, RecNum, TmpPerson
  If NameToSearch=UCase(Trim(TmpPerson.Name)) Then
    Found = True
    Exit For
  End If
Next
```

تستخدم حلقة For العبارة Get، لقراءة سجلات الملف، سجلاً تلو الآخر (تسلسلياً) ونقل محتوياته للمتحول TmpPerson. تُستخدم عبارة If بعد قراءة السجل، للمطابقة بين السجل المقروء وبين الاسم الذي يجري البحث عنه، فتقارن عبارة If بين قيمة المتحول NameToSearch وبين قيمة UCase (Trim(TmpPerson.Name))، يُستخدم التابع الوظيفي UCase() في عملية البحث، حتى لا يكون البحث مقيداً بحالة الحرف (أي سيتم العثور على الاسم المطابق، سواء كان مكتوباً بأحرف كبيرة أم صغيرة)، كما يُستخدم التابع الوظيفي Trim() للتخلص من الفراغات الفائضة، التي قد تكون موجودة قبل أو بعد الاسم في حقل الاسم.

ملاحظة

يُعيد التابع الوظيفى LTrim() السلسلة التى مُررّت إليه، بعد حذف الفراغات الموجودة يسار السلسلة.
يُعيد التابع الوظيفى RTrim() السلسلة بعد حذف الفراغات الموجودة يمينها.
يُعيد التابع الوظيفي Trim() السلسلة بعد حذف الفراغات الموجودة يسارها ويمينها

السلسلة.
فمثلاً لنأخذ النص التالى:

MyVar = "   Testing   "

فإذا نفذّت العبارة التالية:

Result = LTrim(MyVar)

عندها تكون قيمة Result تساوي:

"Testing   "

وإذا نفذّت العبارة التالية:

Result = RTrim(MyVar)

عندها تكون قيمة Result تساوي:

"   Testing"

أما عند تنفيذ العبارة:

Result = Trim(MyVar)

عندها ستكون قيمة Result تساوي:

"Testing"

لاحظ أن العبارات الثلاث التالية متساوية من حيث النتيجة:

Result = LTrim(RTrim(MyVar))
Result = RTrim(LTrim(MyVar))
Result = Trim(MyVar)

تُسند القيمة True إلى المتحول Found إذا تطابق حقل الاسم للسجل مع NameToSearch، ويتم الخروج من الحلقة For. يُظهر الإجراء بعد إنهاء الحلقة For نتائج البحث:

```
If Found = True Then
  SaveCurrentRecord
  gCurrentRecord = RecNum
  ShowCurrentRecord
Else
  MsgBox "لم أجد الاسم:" + NameToSearch
End If
```

فإذا نجحت عملية البحث، يتم حفظ السجل الحالي، وإظهار السجل الذي عُثر عليه. بينما تظهر رسالة تقول بأنه لم يتم العثور على الاسم، الذي طُلب البحث عنه، في حال فشل البحث.

نص الإجراء cmdDelete_Click()

يُنفذ هذا الإجراء عند نقر الزر حذف، ويحذف السجل الحالي. يتخذ الإجراء أربع خطوات لحذف السجل الحالي من الملف Phone.DAT:

- إنشاء ملف مؤقت فارغ باسم Phone.TMP.
- استخدام حلقة For لنسخ كل سجلات الملف Phone.DAT (سجلاً وراء الآخر)، باستثناء السجل الراهن (السجل المراد حذفه) إلى الملف Phone.TMP.
- حذف الملف Phone.DAT بعد نسخ محتوياته للملف المؤقت Phone.TMP.
- إعادة تسمية الملف Phone.TMP بالاسم Phone.DAT.

الملفات ذات الوصول التسلسلي

يتم الوصول إلى الملفات العشوائية سجلاً سجلاً، بينما يتم الوصول إلى الملفات التسلسلية سطراً سطراً، فالبيانات عند كتابتها في ملف تسلسلي، تخزّن كأسطر نصية في الملف.

والحقيقة أن الملفات التسلسلية التى يتم الوصول إليها سطراً سطراً، تكون مثالية للاستخدام في التطبيقات التي تعالج الملفات النصية.

يمكنك فتح ملف تسلسلي من أجل:

- الخرج Output (الكتابة إليه).
- الإضافة Append (إضافة أسطر في نهاية الملف).
- الدخل Input (القراءة من الملف).

فتح ملف من أجل الخرج Output

يلزمك لإنشاء ملف تسلسلي، فتح ملف من أجل الخرج Output.

تستطيع بعد إنشاء الملف، استخدام أوامر الخرج، لكتابة الأسطر في الملف.

يشرح جزء البرنامج النموذجي التالي، كيفية إنشاء الملف TRY.TXT:

```
FileNum = FreeFile
```

```
Open "TRY.TXT" For Output As FileNum
```

يُنشئ جزء البرنامج هذا، الملف TRY.TXT في حال عدم وجوده، أو يتسبب بحذف الملف القديم (في حال وجوده)، وإنشاء ملف جديد مكانه. (لذلك يجب الانتباه في هذه الحالة).

لاحظ أننا لم نحدد مساراً (Path) للملف TRY.TXT في العبارة Open، مما يعني أن الملف سيفتح في المسار الراهن. (موقع الملف التنفيذي).

لنفرض أن المسار الراهن هو C:\PROGRAMS، تُنجز العبارتان التاليتان نفس العمل:

```
Open "TRY.TXT" For Output As FileNum
Open "C:\PROGRAMS\TRY.TXT" For Output As FileNum
```

الملف الذي تم إنشاؤه عند الفتح يكون فارغاً تماماً، تُستخدم العبارة # Print، لكتابة سطر نصي في الملف.

ينشئ المثال التالي، الملف TRY.TXT ويكتب محتويات مربع النص المدعو txtMyText في الملف:

```
FileNum = FreeFile
Open "TRY.TXT" For Output As FileNum
Print #FileNum,TxtMyText.Text
Close FileNum
```

إذاً، تم عبر المثال السابق، تمرير وسيطين إلى العبارة # Print:

- الوسيط الأول: رقم الملف أو مقبضه.
- الوسيط الثاني: السطر المطلوب كتابته في الملف.

فتح ملف تسلسلي للإضافة Append

يتشابه فتح ملف من أجل الإضافة، مع فتحه من أجل الخرج، إلا أن الفرق بينهما، أن فتح ملف من أجل الإضافة، لا يتسبب بحذفه إذا كان موجوداً أصلاً، ولكن يضيف الأسطر إلى آخر الملف المفتوح. لنفترض مثلاً أن الملف TRY.TXT موجود مسبقاً ويحتوي على السطرين التاليين:

```
THIS IS LINE NUMBER 1
THIS IS LINE NUMBER 2
```

لإضافة سطر جديد إلى الملف TRY.TXT نستطيع استخدام جزء البرنامج التالي:

```
FileNum = FreeFile
Open "TRY.TXT" For Append As FileNum
Print #FileNum, "THIS IS A NEW TEXT"
Close FileNum
```

سيحتوي الملف بعد تنفيذ هذا الجزء من البرنامج، على السطور الثلاثة التالية:

```
THIS IS LINE NUMBER 1
THIS IS LINE NUMBER 2
THIS IS A NEW TEXT
```

وإذا أعدت تنفيذ نفس الجزء من البرنامج السابق، فسوف يحتوي الملف على أربعة أسطر:

```
THIS IS LINE NUMBER 1
THIS IS LINE NUMBER 2
THIS IS A NEW TEXT
THIS IS A NEW TEXT
```

فتح ملف تسلسلي من أجل الإدخال

تحتاج عندما ترغب بالقراءة من ملف تسلسلي، إلى فتحه من أجل الإدخال، وحالما يُفتح الملف، يصبح بالوسع استخدام التابع الوظيفي Input() لقراءة كامل محتويات الملف إلى مربع النص txtMyText:

```
FileNum = FreeFile
Open "TRY.TXT" For Input As FileNum
TxtMyText.Text = Input(LOF(FileNum), FileNUm)
Close FileNum
```

يأخذ التابع الوظيفي Input() وسيطين:

- يحدد الوسيط الأول عدد البايتات المطلوب قراءتها من الملف.
- يحدد الوسيط الثاني رقم الملف أو مقبضه.

وباعتبار أن هدف جزء البرنامج هذا، هو قراءة كامل محتويات الملف، فقد حُدد الوسيط الأول للتابع Input() بالشكل LOF(FileNum). حيث يعيد التابع LOF() الطول الكلي للملف مقدراً بالبايتات Bytes.

العبارتان # Write و # Input

استخدِم التابع الوظيفي ()Input والعبارة # Print، في الأمثلة السابقة، لقراءة البيانات من ملف تسلسلي، وكتابة البيانات في ملف تسلسلي. يمكننا استخدام العبارتين # Write و # Input أيضا لمثل هذه الغاية (للكتابة # Write و للقراءة # Input).

تمكّنك العبارة # Write من كتابة لائحة من المتحولات (كتابية أو عددية) في ملف. ينشئ المثال اللاحق الملف TRY.TXT ويحفظ محتويات المتحول الكتابي MyString ومحتويات المتحول العددي MyNumber في الملف:

```
FileNum = FreeFile
Open "TRY.TXT" For Output As FileNum
Write #FileNum, MyString, MyNumber
Close FileNum
```

الوسيط الأول في عبارة # Write هو رقم الملف، وباقي الوسائط هي المتحولات التي ستكتب في الملف. يوجد هنا وسيطان فقط، بعد الوسيط الأول، وذلك لأننا نرغب بكتابة متحولين فقط في الملف.

تمكّنك عبارة # Input من قراءة بيانات من ملف يحوي لائحة من المتحولات، يقرأ النص التالي محتويات الملف الذي أنشأناه للتو:

```
FileNum = FreeFile
Open "TRY.TXT" For Input As FileNum
Input #FileNum, MyString, MyNumber
Close FileNum
```

الوسيط الأول في عبارة # Input هو رقم الملف، أما باقي الوسائط فهي المتحولات المطلوب قراءتها من الملف (أي إسناد قيمها من محتويات الملف).

يجب أن يتطابق ترتيب المتحولات في عبارة # Input، مع الترتيب الذي خُزّنت فيه أصلاً في الملف بواسطة العبارة # Write.

الملفات ذات الوصول الثنائي

يكون الوصول إلى الملفات العشوائية سجلاً سجلاً، ويكون في الملفات التسلسلية سطرً سطراً، بينما يتم الوصول إلى الملفات الثنائية Binary Files بايتاً بايتاً. تستطيع القراءة

من أي موقع من الملف الثنائي حال فتحه من أجل الوصول الثنائي. يُحقّق الوصول الثنائي قدراً كبيراً من المرونة بسبب القدرة على الوصول إلى أي بايت مطلوب في الملف.

فتح ملف من أجل الوصول الثنائي

ينبغي فتح الملف من أجل الوصول الثنائي، قبل التمكن من الوصول إليه بايتاً بايتاً. يفتح جزء البرنامج التالي الملف من أجل الوصول الثنائي:

```
FileNum = FreeFile
Open "TRY.DAT" For Binary As FileNum
```

تُنشئ العبارة السابقة الملف TRY.DAT في حال عدم وجوده.

كتابة بايتات في الملف الثنائي

يمكن استخدام العبارة # Put (بعد فتح ملف من أجل الوصول الثنائي)، لكتابة بايتات في أي موقع من الملف، يكتب جزء البرنامج النموذجي اللاحق السلسلة THIS IS A TEST في الملف TRY.DAT بدءاً من الموقع 100:

```
MyString = "THIS IS A TEST"
FileNum = FreeFile
Open "TRY.DAT" For Binary As FileNum
Put #FileNum, 100, MyString
Close FileNum
```

تأخذ عبارة # Put ثلاثة وسائط:

- يمثل الوسيط الأول رقم الملف أو مقبضه.
- يمثل الوسيط الثاني موقع البايت الذي ستبدأ الكتابة فيه.
- يمثل الوسيط الثالث اسم المتحول الذي ستكتب محتوياته في الملف.

في مثالنا السابق، المتحول الذي ستكتب محتوياته في الملف، عبارة عن متحول كتابي String Variable، وطبعاً يمكن استخدام أنواع متحولات أخرى ولاسيما المتحولات العددية Numeric Variable.

ينتقل الموقع بعد تنفيذه عبارة # Put إلى البايت الذي يلي آخر بايت مكتوب للتو.

فمثلاً، يتحرك الموقع بعد كتابة محتويات المتحول الكتابي، بدءاً من الموقع 100 في مثالنا السابق، إلى الموقع 114 تلقائياً، وعند تنفيذ عبارة # Put أخرى، دون تحديد موقع البداية، فسوف تبدأ عبارة # Put الجديدة من الموقع 114.

يوضح جزء البرنامج التالي كيف يحدّد الموقع ضمن الملف تلقائياً، بعد عبارة # Put:

```
FileNum = FreeFile
Open "TRY.DAT" For Binary As FileNum
MyString = "12345"
Put #FileNum, 20, MyString

MyString = "67890"
Put #FileNum, , MyString
Close FileNum
```

لاحظ أن عبارة # Put الثانية، لم تحدّد موقع بايت بدء الكتابة (أي لم يحدّد الوسيط الثاني، بل وضع فراغ بين أول فاصلة والفاصلة الثانية).

قراءة بايتات من الملف الثنائي

يمكن استخدام العبارة # Get بعد فتح ملف للوصول الثنائي، وذلك لقراءة بايتات من أي موقع بايت ضمن الملف. يقرأ جزء البرنامج النموذجي التالي 15 رمزاً بدءاً من البايت ذو الموقع 40 للملف TRY.DAT:

```
FileNum = FreeFile
Open "TRY.DAT" For Binary As FileNum
MyString = String(15, " ")
Get #FileNum, 40, MyString
Close FileNum
```

تأخذ عبارة # Get ثلاثة وسائط أيضاً:

- يمثل الوسيط الأول رقم الملف أو مقبضه.
- يمثل الوسيط الثاني موقع بايت بداية القراءة.
- يمثل الوسيط الثالث اسم المتحول الذي سيُملأ بالبيانات المقروءة من الملف.

يتحدد عدد البايتات المقروءة من الملف بحجم المتحول الذي سيُملأ بالبيانات، فالمتحول MyString ملىء بـــ 15 رمز فراغ مكرر،وبالتالي يبلغ طوله 15 رمزاً، وهي عدد الأحرف المقروءة من الملف الثنائي.

الخلاصة

تعلمنا كيفية إنشاء الملفات وكتابة البيانات فيها وقراءتها منها أيضاً. كما تعلمنا أن هناك أنواع ثلاثة للملفات، الملفات العشوائية التى تقرأ وتكتب سجلات من البيانات، وهى مناسبة لتطبيقات وقواعد البيانات البسيطة نوعاً ما، والملفات التسلسلية التى تقرأ وتكتب البيانات بشكل أسطر وتعتبر مناسبة للملفات النصية، أخيراً هناك الملفات الثنائية التى تمتاز بالمرونة الكبيرة، كونها تصل بايتا بايتاً إلى الملفات.

الفصل الخامس عشر

المصفوفات، ربط وإدراج كائن ومواضيع أخرى

سنتعلم في هذا الفصل بعض المواضيع المتنوعة في فيجول بيسك، والتي لم تغطها الفصول السابقة.

الملفات من النوع آسكي ASCII

ربما يحتاج المرء عند تطوير البرامج إلى طباعة نسخة مطابقة عن جدول خصائص النموذج ونص البرنامج، يحفظ فيجول بيسك جدول خصائص النموذج، ونص البرنامج، في ملف النموذج. فمثلاً، يحتوي الملف MYFORM.FRM على جدول خصائص النموذج FrmMyForm، ونصوص برنامجه أيضاً.

لنلقِ نظرة على الملف OPTION.FRM الذي طورناه في الفصل الثاني "الخصائص وعناصر التحكم والكائنات".

- شغّل محرر نصوص ما، له القدرة على تحميل ملفات من نوع ASCII (مثل المفكرة Notepad أو الدفتر Word Pad).

افترضنا عبر المناقشة التالية أننا استخدمنا المفكرة، كمحرر للنصوص (تستطيع استخدام برنامج Word إن شئت، لتحميل الملفات النصية).

- اختر **فتح** من قائمة **ملف** للمفكرة، وحمل الملف C:\VB5Prg\Ch02\OPTION.FRM.

يُفترض أن يظهر الملف OPTION.FRM بشكل مشابه لما يلي:

```
VERSION 5.00
Begin VB.Form frmOption
   BackColor       =   &H000000FF&
   Caption         =   "برنامج الخيارات"
   ClientHeight    =   4350
   ClientLeft      =   2400
   ClientTop       =   1620
   ClientWidth     =   4455
   LinkTopic       =   "Form1"
   RightToLeft     =   -1  'True
   ScaleHeight     =   4350
   ScaleWidth      =   4455
   Begin VB.CommandButton cmdExit
      Caption         =   "&خروج"
      Height          =   495
      Left            =   1560
      RightToLeft     =   -1  'True
      TabIndex        =   6
      Top             =   3720
      Width           =   1215
   End
   Begin VB.OptionButton optLevel3
      Alignment       =   1  'Right Justify
      BackColor       =   &H000000FF&
      Caption         =   "المستوى &3"
```

```
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   178
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00FFFFFF&
      Height          =   385
      Left            =   600
      RightToLeft     =   -1  'True
      TabIndex        =   5
      Top             =   1680
      Width           =   1215
   End
   Begin VB.OptionButton optLevel2
      Alignment       =   1  'Right Justify
      BackColor       =   &H000000FF&
      Caption         =   "المستوى &2"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   178
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00FFFFFF&
      Height          =   385
      Left            =   600
      RightToLeft     =   -1  'True
      TabIndex        =   4
      Top             =   960
      Width           =   1215
   End
   Begin VB.OptionButton optLevel1
```

```
      Alignment       =   1  'Right Justify
      BackColor       =   &H000000FF&
      Caption         =   "المستوى &1"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   178
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00FFFFFF&
      Height          =   385
      Left            =   600
      RightToLeft     =   -1  'True
      TabIndex        =   3
      Top             =   240
      Width           =   1215
   End
   Begin VB.CheckBox chkColors
      Alignment       =   1  'Right Justify
      BackColor       =   &H000000FF&
      Caption         =   "الأ&لوان"
      ForeColor       =   &H00FFFFFF&
      Height          =   385
      Left            =   2640
      RightToLeft     =   -1  'True
      TabIndex        =   2
      Top             =   1680
      Width           =   1215
   End
   Begin VB.CheckBox chkMouse
      Alignment       =   1  'Right Justify
      BackColor       =   &H000000FF&
      Caption         =   "ال&فأرة"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
```

```
         Charset         =   178
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00FFFFFF&
      Height          =   385
      Left            =   2640
      RightToLeft     =   -1  'True
      TabIndex        =   1
      Top             =   960
      Width           =   1215
   End
   Begin VB.CheckBox chkSound
      Alignment       =   1  'Right Justify
      BackColor       =   &H000000FF&
      Caption         =   "أ&صوات"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   178
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00FFFFFF&
      Height          =   385
      Left            =   2640
      RightToLeft     =   -1  'True
      TabIndex        =   0
      Top             =   240
      Width           =   1215
   End
   Begin VB.Label lblChoice
      Alignment       =   1  'Right Justify
      BorderStyle     =   1  'Fixed Single
      BeginProperty Font
```

```
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   178
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00000080&
      Height          =   1335
      Left            =   600
      RightToLeft     =   -1  'True
      TabIndex        =   7
      Top             =   2160
      Width           =   3255
   End
End
Attribute VB_Name = "frmOption"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
Private Sub chkColors_Click()
  Updatelabel
End Sub
Private Sub chkMouse_Click()
  Updatelabel
End Sub
Private Sub chkSound_Click()
  Updatelabel
End Sub
Private Sub cmdExit_Click()
  End
End Sub
Private Sub optLevel1_Click()
  Updatelabel
End Sub
Private Sub optLevel2_Click()
  Updatelabel
```

```
End Sub
Private Sub OptLevel3_Click()
  Updatelabel
End Sub
Public Sub Updatelabel()
  Dim Info
  Dim LFCR
  LFCR = Chr(13) + Chr(10)
  ' الصوت
  If chkSound.Value = 1 Then
     Info = "الصوت: تشغيل"
  Else
     Info = "الصوت: إيقاف"
  End If
  ' الفأرة
  If chkMouse.Value = 1 Then
     Info = Info + LFCR + "الفأرة: تشغيل"
  Else
     Info = Info + LFCR + "الفأرة: إيقاف"
  End If
  ' الألوان
  If chkColors.Value = 1 Then
     Info = Info + LFCR + "الألوان: تشغيل"
  Else
     Info = Info + LFCR + "الألوان: إيقاف"
  End If
  ' المستوى 1
  If optLevel1.Value = True Then
     Info = Info + LFCR + "المستوى: 1"
  End If
  'level 2
  If optLevel2.Value = True Then
     Info = Info + LFCR + "المستوى: 2"
  End If
  If optLevel3.Value = True Then
     Info = Info + LFCR + "المستوى: 3"
  End If
  lblChoice.Caption = Info
```

```
End Sub
```

كما تلاحظ، فإن الملف OPTION.FRM عبارة عن ملف آسكي ASCII نظامي (أي ملف نصي عادي). يبين السطر الأول في الملف النصي OPTION.FRM رقم إصدار فيجول بيسك المستخدم لتوليد الملف: Version 5.00، تصف السطور التي تلي هذا السطر، النموذج وعناصر التحكم الموجودة عليه (أي بمعنى آخر، جدول خصائص النموذج). أما الأسطر التي تلي جدول خصائص النموذج، فتمثل نص البرنامج.
يعتبر فهم صيغة الملفات النصية (ملفات ASCII) هام جداً للمبرمجين الذين يرغبون بكتابة برامج تولد ملفات ASCII آلياً، (كأن تكتب برنامجاً يسأل المستخدم بضعة أسئلة، وبناء على أجوبة المستخدم، يولّد البرنامج الملف ASCII للنموذج). لا يمتلك فيجول بيسك أي طريقة تمكّنه من معرفة من أين جاء الملف ASCII، لهذا يجب أن يكون الملف مسايراً بإحكام للصيغة التي يتوقعها فيجول بيسك.

المصفوفات

يتوجب على البرامج التصريح عن المصفوفات إذا أردت استخدامها. يحدّد التصريح عن المصفوفة، اسم المصفوفة وعدد البنود التي يمكنها الاحتفاظ بها. كما أن التصريح عن مصفوفة ضمن قسم التصاريح العامة يجعلها مرئية لكل إجراءات وتوابع النموذج. بينما تكون المصفوفة مرئية لكل نماذج المشروع إذا تم التصريح عنها في وحدة نمطية مستقلة (Module BAS)، وأسبقنا عبارة التصريح بالكلمة المحجوزة Global.

أنواع البيانات Data Type

يدعم فيجول بيسك أنواعاً عديدة من البيانات، يستعرض الجدول 15-1 شتى أنواع البيانات ومجالات قيم البيانات المتوفرة في فيجول بيسك.
لنأخذ مثلاً العبارة التالية للتصريح عن متحول بأنه من النوع الصحيح:

```
Dim Counter As Integer
```

وهي تقابل العبارة التالية (أي تكافئ) العبارة السابقة:

```
Dim Counter%
```

يعتبر الرمز % التمثيل المختصر للعبارة As Integer، يأخذ المتحول من النوع الصحيح، قيمة ما من ضمن المجال –32768 وحتى 32768. تشغل كل قيمة صحيحة 2 Byte (بايتين) من الذاكرة.

بشكل مشابه، يمكننا التصريح عن متحول بأنه صحيح طويل كما يلي:

```
Dim Number As Long
```

تعطي العبارة التالية شكلاً آخر للتصريح عن المتحول Number بأنه من النوع الطويل:

```
Dim Number&
```

الرمز & هو التمثيل المختصر للعبارة As Long. يأخذ المتحول من النوع الصحيح الطويل، أي قيمة من ضمن المجال –2,147,483,648 وحتى 2,147,483,648. يشغل كل متحول طويل 4 Byte (4 بايتات) من الذاكرة.

يصرح عن المتحول المضاعف الدقة بشكل مشابه لما يلي:

```
Dim MyVariable As Double
```

أو يمكن اختصار التصريح السابق بالشكل التالي:

```
Dim MyVariable#
```

المتحول المضاعف الدقة، قد يكون رقماً موجباً أو سالباً ويشغل 8 Byte من الذاكرة.

يوضح الجدول 15–1 مجالات القيم السالبة والموجبة لكل أنواع البيانات في فيجول بيسك.

الجدول 15–1. أنواع البيانات التي يدعمها فيجول بيسك.

نوع البيانات	البايتات المطلوبة	المجال	الاختصار
Integer	2 بايت	–32.768 إلى 32.767	%
Long	4 بايت	–2.147.483.648 إلى 2.147.483.647	&
Single-positive	4 بايت	1.401298E–45 إلى 3.402823E38	!
Single-negative	4 بايت	–3.402823E38 إلى –1.401298E–45	!
Double-positive	8 بايت	4.94065645841247D–24 إلى 1.79769313486232D308	#

Double-negative	8 بايت	1.79769313486232D308− إلى −4.94065645841247D−24	#
Currency	8 بايت	922337203685477,5808− إلى 922337203685477,5807	@
String	يعتمد على عدد الأحرف من صفر إلى حوالي 2 بليون حرف		$
Date	8 بايت	1/جونيه/100 إلى 31/ديسمبر/9999	
Variant	يعتمد على نوع البيانات التي يخزنها		

لاحظ أن آخر عنصر في الجدول 15−1 يصف نوع متحول غريب بعض الشيء، وهو النوع Variant، يمكن أن يشير هذا النوع إلى تاريخ أو وقت أو سلسلة كتابية أو متحول ذي فاصلة عائمة (مثلاً 1234.5678). فمثلاً عند التصريح عن متحول I بالشكل:

```
Dim I As Integer
```

يُشكّل فيجول بيسك متحولاً يدعى I من نوع صحيح، أما عند التصريح عن متحول I بالشكل:

```
Dim I
```

فسوف يشكل فيجول بيسك المتحول I من نوع متغير (Variant)،أي أن فيجول بيسك لا يعلم بماذا سوف يستخدم المتحول I، هل هو سلسلة أم متحول صحيح، أم متحول طويل، أم أي نوع آخر. سيتمكن فيجول بيسك فيما بعد من تحديد نوع المتحول بالنظر إلى العبارة التي تستخدمه. فمثلاً: يدرك فيجول بيسك لدى استخدامه العبارة التالية:

```
I = "My String"
```

أن المتحول يجب معالجته كسلسلة كتابية، أما إذا صادف فيجول بيسك العبارة التالية:

```
I = 2 + 3
```

فسيدرك عندها أن المتحول يجب معالجته كمتحول من نوع صحيح Integer.

ميزة استخدام متحولات متغيرة (Variant) أنه بإمكان نفس المتحول الخدمة بعدة أنواع معطيات. فمثلاً، لا مانع من ورود العبارات التالية في نفس الإجراء:

```
Dim I
I = 2 + 3
I = "My String"
```

يلعب المتحول I هنا دور متحول صحيح، ومتحول من نوع كتابي (سلسلة) في نفس الوقت.

أما عيب استخدام نوع البيانات Variant، فهي التسبب فى بطيء عمل البرنامج، نسبة إلى البرامج التي تحدد بدقة نوع المتحولات التي تستخدمها.

فمثلاً، تُنفّذ حلقة For أسرع، عندما يصرّح عن متحول عداد الحلقة، بأنه من النوع الصحيح. طبعاً قد يكون متحول عداد الحلقة من نوع طويل، إذا كانت الحلقة ستتكرر أكثر من 32767 مرة، وعند ذلك ستدور الحلقة بشكل أبطأ مما هي عليه، عندما يكون عداد الحلقة من نوع صحيح.

يمكنك استخدام متحول من النوع المتغير Variant كعداد لحلقة For. إلا أن الحلقة ستدور بشكل أبطأ، مما لو كان المتحول من النوع الصحيح Integer أو الطويل Long.

يمكن استخدام عبارة مشابهة لما يلي للتصريح عن عداد حلقة من نوع متغير:

```
Dim HisVariant
```

أو تستطيع استخدام عبارة أكثر وضوحاً:

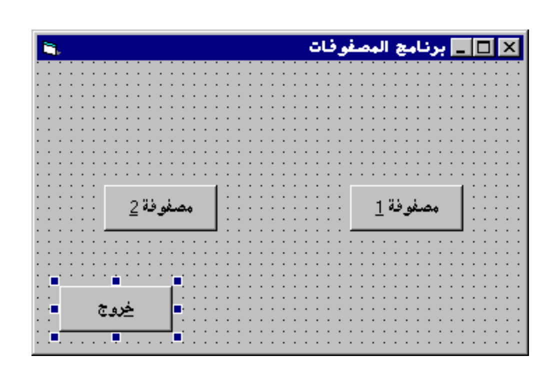
```
Dim HisVariant As Variant
```

برنامج المصفوفات

يوضح برنامج المصفوفات كيفية التصريح عن المصفوفات في فيجول بيسك.

التمثيل المرئي للبرنامج المصفوفات

- أنشئ الدليل C:\VB5Prg\Ch15. سنستخدم هذا الدليل لحفظ العمل المنجز.
- افتح مشروعاً جديداً واحفظ نموذج المشروع بالاسم ARRAYS.FRM في الدليل C:\VB5Prg\Ch15، واحفظ ملف المشروع بالاسم ARRAYS.VBP في ذات الدليل.
- أنشئ النموذج طبقاً للجدول 15-2. النموذج المكتمل سيبدوكما في الشكل15-1.



الشكل 15-1

النموذج frmArrays.

الجدول 15-2. جدول خصائص النموذج FrmArrays

الكائن	الخاصية	القيمة
Form	**Name**	**frmArrays**
	Caption	برنامج المصفوفات
	RightToLeft	True
CommandButton	**Name**	**cmdArray1**
	Caption	مصفوفة &1
	RightToLeft	True
CommandButton	**Name**	**cmdArray2**
	Caption	مصفوفة &2
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True

إدخال نص برنامج المصفوفات

سنبدأ بكتابة نص برنامج المصفوفات:

- أدخل النص التالي في قسم التصاريح العامة للنموذج frmArrays:

```
يجب التصريح عن كل المتحولات
```

```
Option Explicit
```

- أدخل النص التالي ضمن الإجراء cmdArray1_Click() للنموذج frmArrays:

```
Private Sub cmdArray2_Click()
    Dim COUNTER
    Static array1(1 To 10) As String
    array1(1) = "ABC"
    array1(2) = "DEF"
    array1(3) = "GHI"
    frmARRAYs.Cls
    Print "بنود المصفوفة الأولى هي:"
    For COUNTER = 1 To 3 Step 1
      Print "Array1(" + Str(COUNTER) + ")=" + array1(COUNTER(
    Next
End Sub
```

- أدخل النص التالي ضمن الإجراء cmdArray2_Click() للنموذج frmArrays:

```
Private Sub cmdArray2_Click()
    Dim COUNTER
    gArray2(11) = 234
    gArray2(12) = 567
    gArray2(13) = 890
    frmARRAYs.Cls
    Print "بنود المصفوفة الثانية هي:"
    For COUNTER = 11 To 13 Step 1
      Print "gArray2(" + Str(COUNTER) + ")="+ _
             Str (gArray2(COUNTER))
    Next
End Sub
```

- أدخل النص التالي ضمن الإجراء cmdExit_Click() للنموذج frmArrays:

```
Private Sub cmdExit_Click()
    End
End Sub
```

تنفيذ برنامج المصفوفات

- نفّذ برنامج المصفوفات.
- انقر الزر **مصفوفة1**.

يستجيب برنامج المصفوفات، بإظهار بنود المصفوفة الأولى. (انظر الشكل 15-2).

- ❑ انقر الزر **مصفوفة2**.

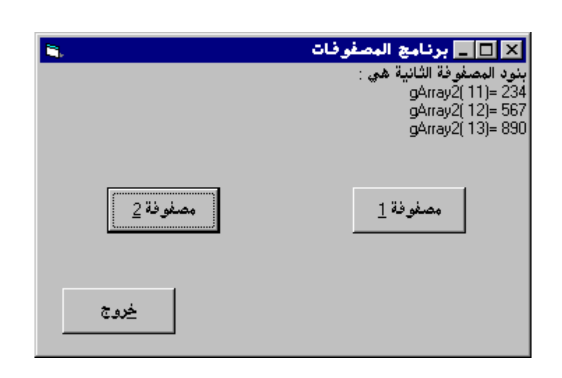
يستجيب برنامج المصفوفات، بإظهار بنود المصفوفة الثانية.

- ❑ انقر الزر **خروج** لإنهاء برنامج المصفوفات.



الشكل 15-2

إظهار بنود Array1().



الشكل 15-3

إظهار بنود Array2().

كيف يعمل برنامج المصفوفات

يُصرّح برنامج المصفوفات عن مصفوفتين أحاديتين Array1() و Array2(). يملأ البرنامج أول ثلاثة بنود من المصفوفة Array1() عند نقر الزر **مصفوفة1**، ويُظهر هذه البنود الثلاثة.

ويملأ البرنامج أيضاً ثلاث بنود من المصفوفة الأحادية Array2()، ويُظهرها عند نقر الزر **مصفوفة2**.

نص قسم التصاريح العامة

يصرّح في هذا القسم عن المصفوفة الأحادية gArray2():

```
Dim gArray2(10 to 20) As Integer
```

هذه المصفوفة من النوع الصحيح (As Integer)، مما يعني أن بنودها من النوع الصحيح.

الرقمين ضمن القوسين الهلاليين، يدلان أن أول بند في هذه المصفوفة هو gArray2(10)، وبالتالي فالبند الثاني هو gArray2(11)، والثالث هو gArray2(12)...الخ. أما آخر بند فهو gArray2(20).

يعني التصريح عن مصفوفة ضمن قسم التصاريح العامة، أنها مرئية لكافة إجراءات وتوابع النموذج.

نص الإجراء cmdArray1_Click()

يُنفّذ هذا الإجراء آلياً عند نقر الزر **مصفوفة1**:

```
Private Sub cmdArray1_Click()
    Dim COUNTER
    Static array1(1 To 10) As String
    array1(1) = "ABC"
    array1(2) = "DEF"
    array1(3) = "GHI"
    frmARRAYs.Cls
    Print "بنود المصفوفة الأولى هي:"
    For COUNTER = 1 To 3 Step 1
      Print "Array1(" + Str(COUNTER) + ")=" + array1(COUNTER)
    Next
End Sub
```

يصرّح هذا الإجراء عن المصفوفة Array1() بالشكل التالي:

```
    Static Array1(1 to 10) As String
```

لاحظ أنه يجب استخدام كلمة Static، قبل اسم المصفوفة عند التصريح عنها في قسم التصاريح العامة، أو يصرّح عنها كمصفوفة ساكنة Static ضمن إجراء.

البند الأول في المصفوفة هو Array1(1)، والبند الثاني هو Array1(2) وهكذا، آخر بند في المصفوفة هو Array1(10).

يملأ الإجراء بعد ذلك ثلاثة بنود في المصفوفة، وينظّف النموذج بواسطة الطريقة Cls:

```
    Array1(1)= "ABC"
```

```
Array1(2)= "DEF"
Array1(3)= "GHI"
frmArrays.Cls
```

وأخيراً يُظهِر الإجراء، أول ثلاثة بنود من المصفوفة باستخدام حلقة For:

```
Print "بنود المصفوفة الأولى هي:"
For COUNTER = 1 To 3 Step 1
  Print "Array1(" + Str(COUNTER) + ")=" + array1(COUNTER)
Next
```

نص الإجراء cmdArray2_Clck()

يُنفّذ هذا الإجراء عند نقر الزر **مصفوفة2**:

```
Private Sub cmdArray2_Click()
    Dim COUNTER
    gArray2(11) = 234
    gArray2(12) = 567
    gArray2(13) = 890
    frmARRAYs.Cls
    Print "بنود المصفوفة الثانية هي:"
    For COUNTER = 11 To 13 Step 1
      Print "gArray2(" + Str(COUNTER) + ")="+ _
                 Str (gArray2(COUNTER))
    Next
End Sub
```

يشبه هذا الإجراء سابقه. لكنه يستعمل المصفوفة Array2() بدون أن يصرّح عنها، لأنها مصفوفة عامة لكل الإجراءات، (وذلك بسبب التصريح عنها في قسم التصاريح العامة).

النهايتان العليا والسفلى للمصفوفة

يتحدد البندان الأول و الأخير في المصفوفة حسب ما أوضحه برنامجنا المصفوفات من التصريح عن المصفوفة. فمثلاً: تصرّح العبارة التالية عن مصفوفة من الأرقام من النوع طويل:

```
Dim MyArray(0 to 35) As Long
```

البند الأول (النهاية السفلى) هو MyArray(0)، والبند الأخير هو MyArray(35).

نستطيع اختصار التصريح السابق بالعبارة التالية:

```
Dim MyArray(35) As Long
```

تفسر العبارة من قبل فيجول بيسك بالشكل التالي: البند الأول في المصفوفة هو MyArray(0)، والبند الثاني هو MyArray(1)...الخ، وآخر بند في المصفوفة هو MyArray(35). عيب استخدام الشكل المختصر للتصريح، أن القيمة الافتراضية للنهاية السفلى هي "الصفر" دوماً، أي ينبغي عليك أن تتذكر دائماً أن البند الأول، هو البند ذي الترتيب "صفر"، بينما البند ذي الترتيب 1 هو البند الثاني وهكذا.

حجم المصفوفة

يتحدد أقصى حجم يمكن أن تصله المصفوفة (الحجم الأعظم) تبعاً لنظام التشغيل، فحسب ما تعلم، يستخدم ويندوز ما يدعى بالذاكرة الوهمية أو الافتراضية Virtual Memory. فعندما تستخدم كل الذاكرة RAM المتاحة، يبدأ ويندوز باستخدام القرص الصلب، وكأنه امتداد للذاكرة RAM. عند ذلك يدعى هذا الجزء من القرص الصلب والمستخدم كذاكرة RAM بالذاكرة الوهمية Virtual Memory.

لنفترض مثلاً أن الحاسب PC يملك ذاكرة RAM بحجم 16 ميجا بايت، وأنه استهلكها كلها، يشرع ويندوز باستخدام حيّز من القرص الصلب كذاكرة RAM، وطبعاً لاشك أن هناك فارق كبير في سرعة القراءة من الذاكرة RAM الفعلية وبين الذاكرة الوهمية، فقراءة البيانات من القرص الصلب تكون أبطأ كثيراً من قراءة البيانات من الذاكرة RAM.

لهذا يجب أن تعلم أن ويندوز سيلجأ إلى استخدام القرص الصلب لدى استعمالك لمصفوفات كبيرة تستهلك الذاكرة RAM المتاحة، مما سيتسبب بتدني سرعة عمل البرنامج، وسيعاني برنامجك من مشاكل في الأداء عند استهلاك كامل الذاكرة RAM المتاحة.

المصفوفات متعددة الأبعاد

استخدم برنامج المصفوفات مصفوفة ذات بعد وحيد، تستطيع في فيجول بيسك التصريح عن مصفوفات متعددة الأبعاد، فمثلاً تصرّح العبارة التالية عن مصفوفة ذات بعدين (ثنائية البعد):

```
Static MyArray(0 to 3, 1 to 4)
```

يمكن سرد بنود هذه المصفوفة بالشكل التالي:

```
MyArray(0,1) MyArray(0,2) MyArray(0,3) MyArray(0,4)
MyArray(1,1) MyArray(1,2) MyArray(1,3) MyArray(1,4)
MyArray(2,1) MyArray(2,2) MyArray(2,3) MyArray(2,4)
MyArray(3,1) MyArray(3,2) MyArray(3,3) MyArray(3,4)
```

تستطيع بشكل مشابه، التصريح عن مصفوفة ثلاثية الأبعاد كما يلي:

```
Dim MyArray(1 to 3, 1 to 7, 1 to 5)
```

يستخدم نص التصريح التالي حلقتي For لملء بنود مصفوفة ثنائية البعد بالقيمة 3:

```
Static MyArray ( 1 To 10, 1 To 10)
Dim COUNTER1, COUNTER2
For COUNTER1 = 1 To 10
  For COUNTER2 = 1 To 10
    MyArray(Counter1, Counter2) = 3
  Next Counter2
Next Counter1
```

المصفوفات الديناميكية

ينبغي الانتباه إلى موضوع الذاكرة عند التصريح عن المصفوفات. فمثلاً، تصرح العبارة التالية عن مصفوفة ذات 10001 بند:

```
Static MyArray(10000) As long
```

كل بند في هذه المصفوفة من النوع Long (طويل)، وبالتالي يشغل 4 Byte، وهذه المصفوفة سوف تشغل 40,004 بايتاً من الذاكرة. لعل هذا القدر ليس بالشيء الكثير، لكن ماذا لو كان لدينا عشر مصفوفات مثل هذه المصفوفة في برنامجنا، ستستهلك المصفوفات العشرة هذه 400,040 بايت من الذاكرة.! لهذا حاول دائماً حجز أقل قدر ممكن للمصفوفة في البرنامج.

لكن هناك ملاحظة أخرى، فبعض المصفوفات يتحدد حجمها فقط أثناء زمن التنفيذ، يمكن في هذه الحالة استخدام العبارة ReDim (إعادة الحجز).

تدعى المصفوفة التي تغير حجمها أثناء زمن التنفيذ بالمصفوفة الديناميكية Dynamic Array.

يوضح جزء البرنامج التالي طريقة إعادة حجز مصفوفة:

```
Private Sub Command1_Click()
    Dim Counter
    Dim Array1() As Integer
    ReDim Array1(1 To 15) As Integer
    For Counter = 1 To 15 Step1
      Array1(Counter) = Counter
    Next Counter
    ReDim Array1(1 To 15) As Integer
```

يصرّح الإجراء عن Array1() كمصفوفة ديناميكية، لاحظ الاختلاف بين التصريح عن المصفوفة الثابتة وبين المصفوفة الديناميكية.

العبارة التالية والتي استخدمناها في النص السابق تعين حجم المصفوفة باستخدام العبارة ReDim:

```
    ReDim Array1(1 to 15) As Integer
```

هذه العبارة تحدد حجم المصفوفة بـــ 15 بنداً، وكل بند من بنودها ينتمي إلى النوع الصحيح.

تملأ الحلقة For بنود المصفوفة الخمسة عشر، ثم تستخدم عبارة ReDim مجدداً لإعادة حجز المصفوفة:

```
    ReDim Array1(1 to 5) As Integer
```

بعد تنفيذ العبارة ReDim يكون حجم المصفوفة الثانية فقط خمسة بنود. وهكذا يمكنك استخدام هذه التقنية لتبديل حجم مصفوفاتك أثناء زمن التنفيذ، والحفاظ على أكبر قدر حر من ذاكرة الحاسب. تطلبت Array1() في النص السابق (2×15=30 بايت)، وبعد تنفيذ العبارة ReDim الثانية، يتبدّل حجم Array1() إلى خمسة بنود من النوع الصحيح وبالتالي إلى 10 بايت فقط.

ما يهمنا معرفته، أنه في حال تنفيذ عبارة ReDim، تصبح القيم التي كانت مخزنة في المصفوفة مفقودة إلى الأبد.! فإذا أردت المحافظة على بعض قيم المصفوفة لا بد لك من استخدام الكلمة المحجوزة Preserve، يوضح برنامج المصفوفات2 كيفية إنجاز ذلك.

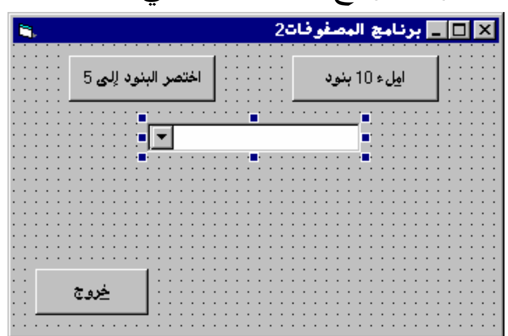
برنامج المصفوفات2

يوضح هذا البرنامج، كيف تتمكن من إعادة حجز المصفوفات، ضمن برامج فيجول بيسك.

التمثيل المرئي لبرنامج المصفوفات2

سنبدأ بتمثيل برنامج المصفوفات2:

- أنشئ مشروعاً جديداً من نوع Standard EXE، واحفظ النموذج الجديد بالاسم ARRAYS2.FRM في الدليل C:\VB5Prg\Ch15، واحفظ ملف المشروع بالاسم ARRAYS2.VBP في الدليل C:\VB5Prg\Ch15.
- أنشئ النموذج frmArray طبقاً للمواصفات المبينة في الجدول 15-3. يُفترض أن يبدو النموذج المكتمل كما في الشكل 15-4.



الشكل 15-4

النموذج frmArray

لبرنامج المصفوفات2.

الجدول 15-3. جدول خصائص النموذج FrmArray.

الكائن	الخاصية	القيمة
Form	**Name**	**frmArray**
	Caption	برنامج المصفوفات2

	RightToLeft	True
CommandButton	**Name**	**cmdFill10**
	Caption	ا&ملء 10 بنود
	RightToLeft	True
CommandButton	**Name**	**cmdOnly5**
	Caption	اختصر البنود إلى 5
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
ComboBox	**Name**	**cboElements**
	Text	(اجعله فارغاً)
	Style	0-Drop Down Combo

إدخال نص برنامج المصفوفات2

- أدخل النص التالي في قسم التصاريح العامة للنموذج frmArray:

```
'يجب التصريح عن كل المتحولات
Option Explicit
Dim gTheArray() As Integer
```

- أدخل النص التالي ضمن الإجراء cmdExit_Click() للنموذج frmArray:

```
Private Sub cmdExit_Click()
    End
End Sub
```

- أدخل النص التالي ضمن الإجراء cmdFill10_Click() للنموذج frmArray:

```
Private Sub cmdFill10_Click()
    ReDim gTheArray(1 To 10) As Integer
    Dim Counter
    For Counter = 1 To 10
      gTheArray(Counter) = Counter
    Next

    cboElements.Clear
```

```
    For Counter = 0 To 9
      cboElements.AddItem Str(gTheArray(Counter + 1))
    Next
End Sub
```

❑ أدخل النص التالي ضمن الإجراء ()cmdOnly5_Click للنموذج frmArray:

```
Private Sub cmdOnly5_Click()
    ReDim Preserve gTheArray(1 To 5) As Integer
    Dim Counter
    cboElements.Clear
    For Counter = 0 To 4
      cboElements.AddItem Str(gTheArray(Counter + 1))
    Next
End Sub
```

تنفيذ برنامج المصفوفات2

❑ نفّذ برنامج المصفوفات2.

❑ انقر الزر **املأ 10 بنود**.

يستجيب برنامج المصفوفات2، بملء مربع التحرير والسرد بعشرة بنود. وتستطيع رؤية هذه البنود بنقر رأس السهم النازل لمربع التحرير والسرد لاستعراض لائحة البنود (انظر الشكل 15-5).

❑ انقر الزر **اختصر البنود إلى 5**

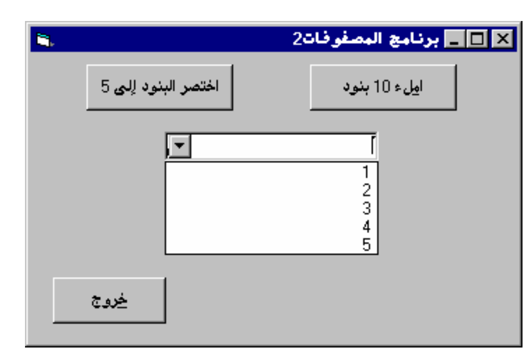
يستجيب برنامج المصفوفات2، بإزالة خمس بنود من مربع التحرير والسرد. يمكنك مشاهدة البنود الخمسة الباقية بنقر رأس السهم النازل لمربع التحرير والسرد لمشاهدة لائحة البنود. (انظر الشكل 15-6).



الشكل 15-5

ملء مربع التحرير والسرد

بعشرة بنود.



الشكل 15-6
الإبقاء على خمسة بنود فقط من المصفوفة.

- انقر الزر **خروج** لإنهاء البرنامج.

كيف يعمل برنامج المصفوفات2

يملأ برنامج المصفوفات2 مصفوفة ما، ويملأ بعدها مربع التحرير والسرد ببنود هذه المصفوفة، يُعاد حجز المصفوفة باستخدام الكلمة المحجوزة Preserve والتي تحافظ على قيم البنود المتبقية في المصفوفة.

نص التصاريح العامة

يصرّح في هذا القسم عن مصفوفة ديناميكية:

```
Dim gTheArray() As Integer
```

طبعاً يفهم فيجول بيسك من هذه العبارة، أن المصفوفة ديناميكية وذلك لأن القوسين الهلاليين وراء الاسم فارغان.

نص الإجراء cmdFill10_Click()

يُنفّذ هذا الإجراء آلياً، عند نقر الزر **املء 10 بنود**:

```
Private Sub cmdFill10_Click()
    ReDim gTheArray(1 To 10) As Integer
    Dim Counter
    For Counter = 1 To 10
      gTheArray(Counter) = Counter
    Next

    cboElements.Clear
```

```
    For Counter = 0 To 9
      cboElements.AddItem Str(gTheArray(Counter + 1))
    Next
End Sub
```

يستخدم الإجراء العبارة ReDim لتحديد حجم المصفوفة بعشرة بنود:

```
    ReDim gTheArray(1 to 10) As Integer
```

تُستخدم حلقة For لملء بنود المصفوفة:

```
    For Counter = 1 To 10
      gTheArray(Counter) = Counter
    Next
```

تُمحى بنود مربع التحرير والسرد، ثم تُستخدم حلقة For لملئه بعشرة بنود، وهذه البنود تتوافق مع بنود المصفوفة (أي يتقابل كل بند من المصفوفة، مع بند من مربع التحرير والسرد):

```
    cboElements.Clear
    For Counter = 0 To 9
      cboElements.AddItem Str(gTheArray(Counter + 1))
    Next
```

لاحظ أن أول بند في مربع التحرير والسرد هو البند ذو الرقم "صفر"، أما أول بند في المصفوفة فرقمه "1"، كذلك آخر بند في مربع التحرير والسرد هو البند رقم "4"، أما آخر بند في المصفوفة فرقمه "10".

نص الإجراء cmdOnly5_Click()

يُنفذ هذا الإجراء آلياً، عند نقر الزر **اختصر البنود إلى 5**:

```
Private Sub cmdOnly5_Click()
    ReDim Preserve gTheArray(1 To 5) As Integer
    Dim Counter
    cboElements.Clear
    For Counter = 0 To 4
      cboElements.AddItem Str(gTheArray(Counter + 1))
    Next
End Sub
```

يستخدم الإجراء عبارة ReDim لتغيير حجم المصفوفة إلى خمسة بنود:

```
    ReDim Preserve gTheArray(1 to 5) As Integer
```

تتسبب الكلمة المحجوزة Preserve بالإبقاء على قيم أول خمسة بنود من المصفوفة على حالها (أي تحتفظ البنود بقيمها الأصلية). يمحو الإجراء بعدها بنود مربع التحرير والسرد ويملؤه بخمسة بنود:

```
cboElements.Clear
For Counter = 0 To 9
  cboElements.AddItem Str(gTheArray(Counter + 1))
Next
```

اتبع الخطوات التالية لرؤية دور الكلمة المحجوزة Preserve:

- احذف الكلمة المحجوزة Preserve من العبارة ReDim لتصبح العبارة بالشكل التالي:

```
ReDim gTheArray(1 to 5) As Integer
```

- نفّذ برنامج المصفوفات2، ثم انقر الزر **املء 10 بنود**، وافتح مربع التحرير والسرد، وتأكد من امتلائه بعشر بنود
- انقر بعد ذلك الزر **اختصر البنود إلى 5** وافتح مربع التحرير والسرد، وتأكد أن اللائحة مملوءة بخمسة بنود، ستجد أن قيم هذه البنود الخمس مساوية كلها إلى الصفر، وذلك بسبب حذف الكلمة المحجوزة Preserve.
- انقر الزر **خروج** لإنهاء برنامج المصفوفات2.

المصفوفات التي يزيد حجمها عن 64KB

يمكن لحجم المصفوفة أن يتجاوز 64KB، والمصفوفة التي تتجاوز 64KB تدعى بالمصفوفة الضخمة Huge Array، ليس هنالك تصريح خاص للمصفوفات الضخمة، فهذه المصفوفات عادية، لكن هناك استثناء وحيد، فإذا كانت المصفوفة الضخمة من نوع String، (أي من نوع نصي)، فيجب أن تكون كل بنودها تحمل نفس العدد من الرموز.

برنامج المغيّر

تستطيع كما تعلم، استخدام الخيار **Add Procedure** من القائمة **Tools** لبناء إجراءات أو توابع وظيفية خاصة بك، وتمتلك هذه الإجراءات أو التوابع وسائط (معاملات) Parameters خاصة بها، وتسمى هذه الوسائط في أدبيات البرمجة، بالمتغيرات الوسيطية Arguments.

هناك طريقتان لتمرير الوسائط Parameters إلى تابع وظيفي في فيجول بيسك:

- طريقة القيمة الراجعة By Reference. (تمرير عنوان المتحول في الذاكرة).
- طريقة القيمة الغير راجعة By Value. (تمرير نسخة عن قيمة المتحول).

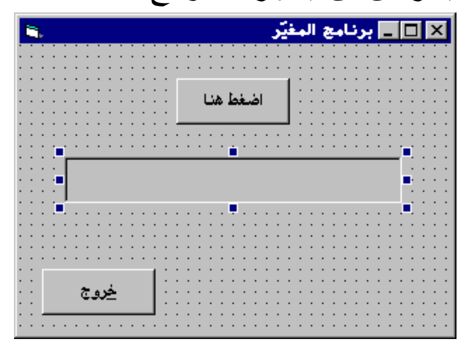
يوضح برنامج المغيّر الاختلاف بين الطريقتين.

التمثيل المرئي لبرنامج المغيّر

□أنشئ مشروعا جديداً من النوع Standard EXE واحفظ نموذج المشروع بالاسم Vary.FRM في الدليل C:\VB5Prg\Ch15 واحفظ ملف المشروع بالاسم VARY.VBP في ذات الدليل.

□ أنشئ النموذج frmVary طبقاً للجدول 15-4.

يفترض أن يظهر النموذج المكتمل كما في الشكل 15-7.



الشكل 15-7

النموذج frmVary.

الجدول 15-4. جدول خصائص النموذج frmVary.

الكائن	الخاصية	القيمة

Form	**Name**	**frmVary**
	Caption	برنامج المغيّر
	RightToLeft	True
CommandButton	**Name**	**cmdDoIt**
	Caption	اضغط هنا
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
Label	**Name**	**lblInfo**
	Caption	(اجعله فارغاً)
	Alignment	2-Center
	BorderStyle	1-Fixed Single

- تأكد من أن قسم التصريحات العامة يحوي العبارة Option Explicit:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أدخل النص التالي ضمن الإجراء cmdDoIt_Click() للنموذج frmVary:

```
Private Sub cmdDoIt_Click()
    Dim V As Integer
    Dim Result As Integer
    V = 3
    Result = VSquare(V)
    lblInfo.Caption = "V = " + Str(V) + " 4*4=" + Str(Result)
End Sub
```

- أدخل النص التالي ضمن الإجراء cmdExit_Click() للنموذج frmVary:

```
Private Sub cmdExit_Click()
    End
End Sub
```

- أنشئ تابعاً وظيفياً جديداً في النموذج frmVary وأطلق عليه التسمية VSquare. وذلك بإظهار نافذة نص البرنامج، واختيار **Add Procedure** من

القائمة **Tools**، ثم تحديد النوع Type على أنه Function، وتحديد الاسم Name بأنه VSquare ثم نقر **OK**.

يستجيب فيجول بيسك بإضافة التابع الوظيفي VSquare().

- غيّر أول سطر بحيث يصبح كما يلي:

```
Public Function VSquare(ByVal V As Integer)

End Function
```

- والآن أدخل النص التالي ضمن التابع الوظيفي VSquare():

```
Public Function VSquare(ByVal V As Integer)
    V = 4
    VSquare = V * V
End Function
```

تنفيذ برنامج المغير

- نفّذ برنامج المغيّر.
- انقر الزر **اضغط هنا**.

يستجيب البرنامج بإظهار القيمة في اللافتة lblInfo كما هو مبين في الشكل 15-8.

- انقر الزر **خروج** لإنهاء البرنامج.



الشكل 15-8

برنامج المغيّر عند تمرير المتحول V إلى التابع الوظيفي VSquare().

كيف يعمل برنامج المغيّر

يمرّر البرنامج متحولاً إلى التابع الوظيفي VSquare() بطريقة القيمة غير الراجعة ByVal، ولذلك لا تتغير قيمة المتحول V من الرقم 3 إلى الرقم 4، كما سترى لاحقاً.

نص الإجراء ()cmdDoIt_Click

يُنفّذ هذا الإجراء آلياً، عند نقر الزر **اضغط هنا:**

```
Private Sub cmdDoIt_Click()
    Dim V As Integer
    Dim Result As Integer
    V = 3
    Result = VSquare(V)
    lblInfo.Caption = "V = " + Str(V) + " 4*4=" + Str(Result)
End Sub
```

يُسند الإجراء القيمة 3 إلى المتحول V، ثم يُنفذ التابع الوظيفي ()VSquare، ويُعيد ناتج ضرب (4×4). يُظهر الإجراء بعد ذلك، قيمة المتحول V والقيمة المعادة من التابع الوظيفي ()VSquare في اللافتة lblInfo.

نص التابع الوظيفي ()VSquare

يُستدعى التابع الوظيفي ()VSquare من قبل الإجراء ()cmdDoIt_Click:

```
Public Function VSquare(ByVal V As Integer)
    V = 4
    VSquare = V * V
End Function
```

يُسند التابع القيمة 4 إلى المتحول V، ثم يُسند القيمة V×V إلى متحول التابع الوظيفي VSquare، الشيء الذي نحب التنبيه إليه، أن المتحول V قد تم تمريره إلى التابع الوظيفي بطريقة القيمة غير الراجعة ByVal، بمعنى أن الإجراء يُنشئ نسخة جديدة عن المتحول المدعو V، ثم يتعامل معه. ولا توجد صلة بين المتحول V في الإجراء طالب الاستدعاء (أي ()cmdDoIt_Click في مثالنا هذا)، وبين المتحول V في التابع الوظيفي ()VSquare.

وهذا يشرح السبب الذي دفعنا إلى إسناد القيمة 3 إلى المتحول V وليس القيمة 4. (انظر إلى الشكل 15-8). فقيمة المتحول الممرر بطريقة القيمة غير الراجعة ByVal. لا تتغير في البرنامج طالب الاستدعاء.

تعديل برنامج المغيّر

سنعدّل الآن برنامج المغيّر، بحيث يبدو كالتالي (لاحظ التعديل في أول سطر من التابع الوظيفي):

```
Public Function VSquare(V As Integer)
    V = 4
    VSquare = V * V
End Function
```

❑ نفّذ برنامج المغيّر، وانقر الزر **اضغط هنا**.

يستجيب البرنامج بإظهار القيم في اللافتة lblInfo حسب ما يتبين من الشكل 15-9.



الشكل 15-9

برنامج المغيّر لدى تمرير المتحول V بطريقة القيمة الراجعة (ByRef).

لاحظ أن المتحول V الآن، لم يمرر بطريقة القيمة غير الراجعة، إلى التابع الوظيفي VSquare()، بل تم تمريره بطريقة القيمة الراجعة، أي أن المتحول الممرر، هو نفس المتحول في الإجراء طالب الاستدعاء، والتابع الوظيفي المستدعى. وهكذا فعند تمرير المتحول V بطريقة القيمة الراجعة، فهذا يعني أن المتحول V في الإجراء cmdDoIt_Click()، هو نفسه في التابع الوظيفي VSquare()، وهذا يشرح سبب ظهور قيمة V في اللافتة lblInfo يحمل القيمة 4 (انظر إلى الشكل 15-9).

يتمكّن المبرمجون بلغة C، من التمييز بين تمرير الوسائط بطريقة القيمة غير الراجعة، وبطريقة القيمة الراجعة، بشكل أكبر، لأنهم يتمكنون من تمييز أن تمرير وسيط بطريقة القيمة الراجعة، يكافئ تمرير عنوان المتحول في الذاكرة. يكفينا هنا أن تلاحظ الفرق بين الطريقتين:

■ عندما يتم تمرير الوسيط بطريقة القيمة غير الراجعة (ByVal). لا يتمكن التابع الوظيفي المستدعى، من تغيير قيمة الوسيط الممرر في الإجراء الطالب

للاستدعاء، (لأنه لا يعرف عنوانه في الذاكرة)، فمثلاً رغم أن التابع VSquare()، يُسند القيمة 4 إلى V، في برنامج المغيّر، إلا أن الإجراء cmdDoIt_Click() يحافظ على قيمة V تساوي 3.

■عندما يتم تمرير الوسيط بطريقة القيمة الراجعة (ByRef). يستخدم كل من الإجراء الطالب للاستدعاء والتابع الوظيفي المستدعى نفس المتحول، (لأن الإجراءين يعرفان عنوان المتحول في الذاكرة)، فمثلاً أدى إسناد القيمة 4 إلى V ضمن التابع الوظيفي VSquare() في النسخة المعدّلة من برنامج المغيّر، إلى تعديل قيمة V في الإجراء cmdDoIt_Click()، فظهرت V تساوي 4 في اللافتة lblInfo.

تعديل برنامج المغيّر مجدداً

لنمضي أكثر في تعديل برنامج المغيّر:

- غيّر VSquare() ليصبح كالتالي:

```
Public Function VSquare(VV As Integer)
    VV = 4
    VSquare = VV * VV
End Function
```

- نفّذ برنامج المغيّر وانقر الزر **اضغط هنا**.

يستجيب برنامج المغيّر مجدداً كما في الشكل 15-9.

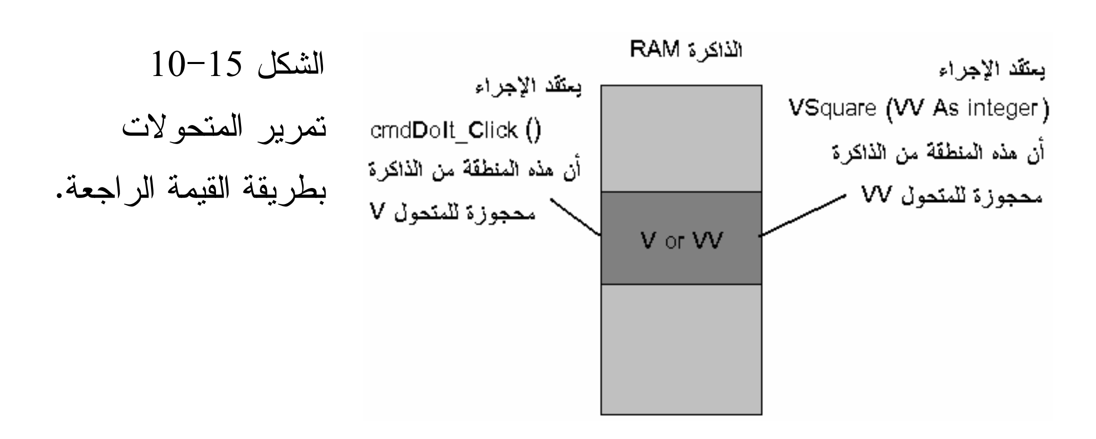
كما تلاحظ، لم يؤثر التغيير في اسم المتحول ضمن VSquare() من V إلى VV على طريقة عمل البرنامج وذلك بسبب تمرير المتحول بطريقة القيمة الراجعة.! وهكذا فرغم أن المتحول يدعى VV ضمن VSquare()، لكنه يبقى نفس المتحول الممرر من قبل الإجراء طالب الاستدعاء (أي أن V و VV هما نفس المتحول).

ملاحظة

انظر الشكل 15-10 لفهم مضمون طريقة تمرير المتحولات بالقيمة الراجعة بشكل أعمق.

حسب الشكل، توجد خلايا من الذاكرة RAM مستخدمة لحفظ المتحول V الخاص بالإجراء ()cmdDoIt_Click، يمرر الإجراء ()cmdDoIt_Click المتحول V إلى التابع ()VSquare بطريقة القيمة الراجعة، ولهذا يستخدم ()VSquare بدوره نفس المتحول باستخدام نفس الخلايا في الذاكرة RAM.
إذاً يستخدم ()cmdDoIt_Click منطقة من الذاكرة RAM لحفظ المتحول V، ويستخدم ()VSquare بدوره نفس المنطقة من الذاكرة RAM لحفظ المتحول VV، وفي كلتا الحالتين فإن V و VV هما نفس المتحول.

الشكل 15−10
تمرير المتحولات
بطريقة القيمة الراجعة.



ربط وإدراج الكائن OLE

يُعتبر موضوع ربط وإدراج الكائن Object Linking and Embedding من مواضيع ويندوز الهامة التي يدعمها فيجول بيسك، وطبعاً تحتاج تغطية هذا الموضوع بشكل مفصل إلى كتاب كامل.
تعتبر خاصية OLE من القوة بحيث يمكن التفكير بها كبرمجة كاملة مستقلة، مدمجة ضمن فيجول بيسك، يعطي ما تبقى من هذا الفصل مقدمة مختصرة عن OLE.

تعريف OLE.؟

هذا المصطلح (OLE) مأخوذ من حروف أولى كلمات الجملة ربط و إدراج كائن (**O**bject **L**inking and **E**mbedding)، يستطيع البرنامج الذي يحوي هذه الخاصية، الاتصال

مع معطيات (نص أو رسوم أو أي معلومات أخرى) من تطبيقات ويندوز الأخرى التي تدعم بدورها هذه الخاصية.

لنفرض مثلاً أن مستخدم برنامجك، يرغب بتعديل ملف ما من نوع BMP، لعل أحد الطرق تتمثل بكتابة برنامج فيجول بيسك يسمح للمستخدم برسم وتعديل الصور النقطية BMP، إلا أن مثل هذا الأمر يلزمه عمل كبير، ويستغرق وقتاً طويلاً حتى تعلّم المستخدم كيف يستخدم برنامج الرسوم الخاص بك لتعديل ملفات الصورة BMP.

لعل الحل الأسهل يتمثل بتحميل الصورة BMP باستخدام برنامج رسوم شهير (مثل الرسامPaint brush)، ثم تعديل الصورة بواسطته، وباعتبار أن المستخدم يعمل حتماً على ويندوز، فلا بد أنه غالباً يعرف كيفية استخدام الرسام Paint؟ كبرنامج رسم مرفق مع ويندوز.

كيف ستخبر المستخدم طريقة استخدام برنامج الرسام Paint؟ تستطيع كتابة برنامج بسيط بلغة فيجول بيسك، يستعرض طريقة تحميل صورة نقطية BMP ثم تعديلها بواسطة الرسام. بشكل خطوات، لكن الطريقة الأفضل والأكثر إحكاماً، تتمثل باستخدام عنصر التحكم OLE (OLE Control) حسب ما يوضحه برنامج الرسام. يوضح الشكل 15-11 رمز عنصر التحكم OLE في إطار أدوات فيجول بيسك.



الشكل 15-11

عنصر التحكم OLE.

برنامج الرسام

يوضح برنامج الرسام كيف يمكن استخدام برنامج يدعم تقنية OLE، مثل برنامج Paint Brush واستخدامه في تطبيقات فيجول بيسك، يمكّنك برنامج الرسام من تحميل صورة نقطية من نوع BMP وتعديلها.

التمثيل المرئي لبرنامج الرسام

- أنشئ مشروعاً جديداً من نوع Standard EXE واحفظ نموذج المشروع بالاسم USEPAINT.FRM في الدليل C:\VB5Prg\Ch15 واحفظ ملف المشروع بالاسم USEPAINT.VBP في ذات الدليل.
- أسند الخصائص التالية للنموذج:

```
Name:          frmUsePaint
Caption:       برنامج الرسام
RightToLeft:   True
```

إضافة عنصر التحكم OLE إلى النموذج frmUsePaint

- انقر نقرة مزدوجة على رمز عنصر التحكم OLE في مربع الأدوات.

يستجيب فيجول بيسك بوضع عنصر التحكم OLE في النموذج وإظهار مربع الحوار إدراج كائن (انظر الشكل 15-12).



الشكل 15-12
مربع الحوار
إدراج كائن.

يُظهر مربع الحوار إدراج كائن لائحة من الكائنات التي يمكن إدراجها في برنامج الرسام.

- اختر **صورة نقطية** من لائحة الكائنات (Objects) ثم انقر الزر **موافق**.
- احفظ المشروع.

تنفيذ البرنامج الرسام

لننفذ برنامج الرسام ونشاهد تقنية OLE قيد التنفيذ:

- نفّذ برنامج الرسام.

يظهر برنامج الرسام كما في الشكل 15-13.

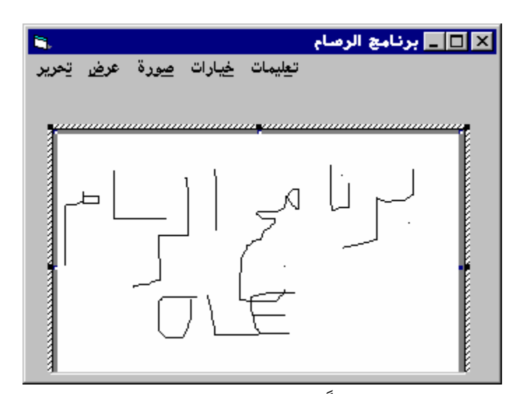
- انقر نقراً مزدوجاً على عنصر التحكم OLE ضمن إطار برنامج الرسام.

يستجيب البرنامج بتمكينك من الرسم ضمن عنصر التحكم OLE، بنفس الأسلوب الذي ترسم به ضمن الرسام (انظر الشكل 15-14). لاحظ أنك تستطيع إنجاز مهام الرسام الأخرى باستعمال القوائم التي تظهر في قمة إطار برنامج الرسام.



الشكل 15-13

برنامج الرسام بعد تنفيذه.



الشكل 15-14

الرسم ضمن عنصر

التحكم OLE.

- ارسم شيئاً ما بواسطة برنامج الرسام
- انقر الرمز (×) في أعلى اليمين لإنهاء برنامج الرسام.

الخلاصة

تعلمنا في هذا الفصل كيف نقرأ ونعدل الملفات من نوع FRM التي تحوي وصف النموذج وفق صيغة ASCII. وتعلمنا كيف نصرح عن المصفوفات الساكنة

والديناميكية، وكيف نستخدمها. وتعلمنا أيضاً كيفية تمرير الوسائط بطريقة القيمة الراجعة ByRef، أو بطريقة القيمة الغير راجعة ByVal، والفرق بين الطريقتين.

كما أعطانا الفصل مقدمة تمهيدية عن تقنية OLE، وكيفية إستخدامها فى تطبيقات فيجول بيسك المختلفة.

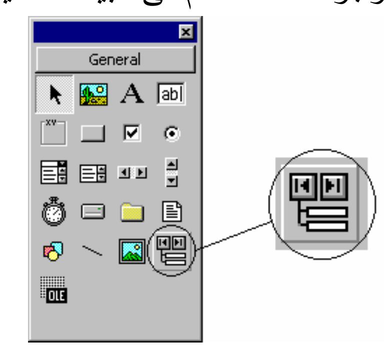
الفصل السادس عشر

أداة التحكم فى البيانات ولغة SQL

يوضح هذا الفصل كيفية استخدام أداة التحكم فى البيانات Data Control، المضمنة مع اللغة الأصلية. تستطيع مع هذه الأداة، كتابة برامج تتيح لك الوصول لقواعد البيانات المختلفة، مثل Microsoft Access، DBase، Btrieve، Paradox، FoxPro. يشرح لك هذا الفصل أيضاً، كيفية استخدام لغة الاستعلامات البنيوية SQL لمعالجة قواعد البيانات ببراعة وسهولة.

برنامج البيانات

يوضح برنامج البيانات، سهولة الوصول للبيانات الموجودة في قواعد البيانات المختلفة، وذلك من خلال فيجول بيسك نفسه، واستخدام أداة التحكم فى البيانات. يوضح الشكل 16-1 رمز أداة التحكم فى البيانات الموجود في نافذة مربع الأدوات، قد يظهر رمز أداة التحكم فى البيانات في موقع مختلف في نافذة الأدوات لديك، عن الشكل 16-1، عند وضع مؤشر الفأرة فوق رمز أداة التحكم فى البيانات في نافذة مربع الأدوات، يظهر مستطيل أصفر اللون يحتوي الكلمة Data. عندها ستتأكد من وجود أداة التحكم فى البيانات لديك.



الشكل 16-1

أداة التحكم فى البيانات.

إنشاء قاعدة بيانات وجداول التحكم فى البيانات

يتصل برنامج البيانات الذي ستصممه لاحقاً، بقاعدة بيانات. وهي ضرورية لعمل البرنامج، لذلك يجب أولاً إنشاء ملف قاعدة البيانات. ستنشئ في هذا الفصل قاعدة

بيانات، باستخدام برنامج خدمي يسمى مدير التحكم فى البيانات Data Manager، الموجود في لغة فيجول بيسك.

سنستخدم برنامج مدير التحكم فى البيانات، لتصميم وإنشاء قاعدة بيانات تسمى Test.mdb، وهي عبارة عن مجموعة من الجداول، مثال، يمكن أن تحتوي الشركة على الجداول التالية:

جدول يضم لائحة أقسام الشركة الصناعية.

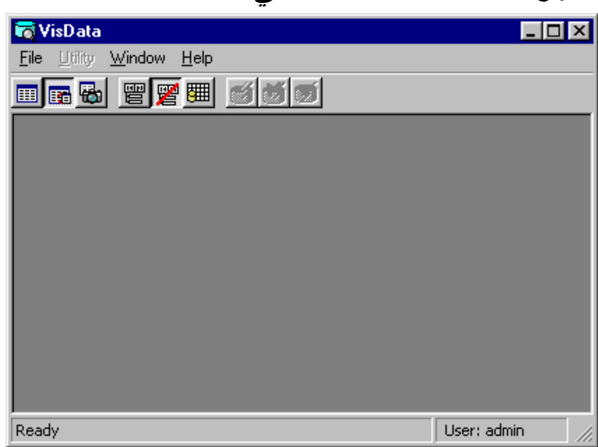
جدول يضم لائحة أسماء بائعي منتجات الشركة.

جدول يضم لائحة أسماء الزبائن.

سوف تضم قاعدة البيانات Test.MDB، جدولاً واحداً هو Parts.

- أنشئ الدليل C:\Vb5Prg\Ch16 لتحفظ عملك فيه.
- اختر البند **Visual Data Manager**، من القائمة **Add Ins**.
- قد يسألك فيجول بيسك إذا كنت تريد إضافة?System.Md للملفini، انقر الزر **لا**.

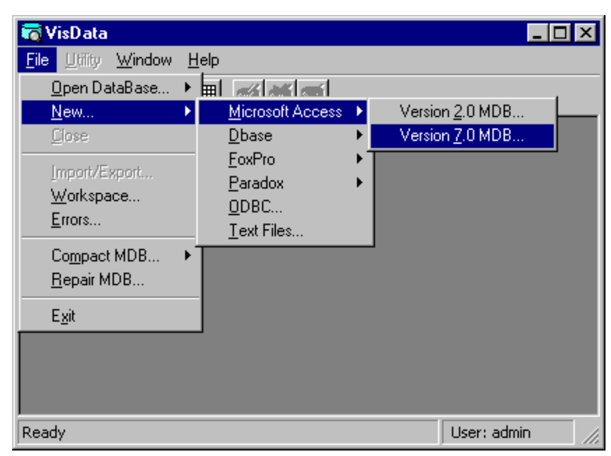
تظهر نافذة VisData كما في الشكل 16-2.

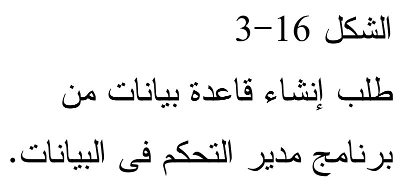


الشكل 16-2

برنامج مدير التحكم فى البيانات.

- اختر البند **New**، ثم البند **Microsoft Access**، ثم البند **Version 7.0 MDB**، من القائمة **File**، انظر الشكل 16-3.

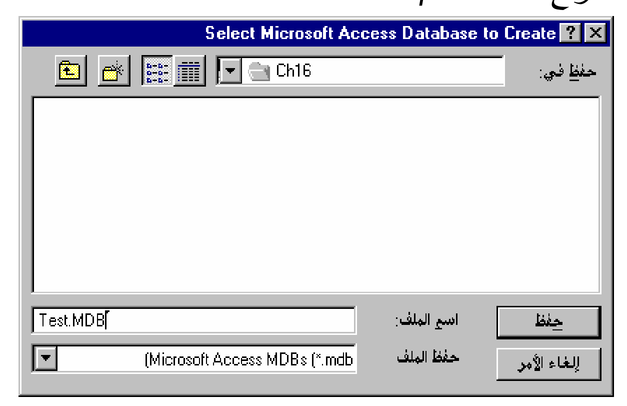




الشكل 3-16

طلب إنشاء قاعدة بيانات من برنامج مدير التحكم فى البيانات.

يظهر عندها مربع حوار (انظر الشكل 16-4)، يسمح لك بإنشاء قاعدة بيانات من النوع Microsoft Access.



الشكل 4-16

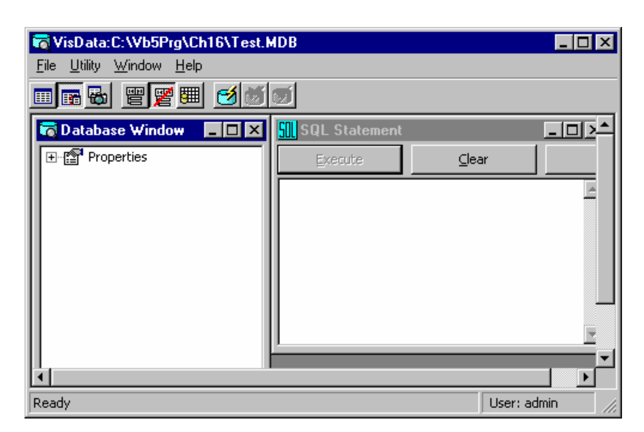
مربع حوار إنشاء ملف قاعدة بيانات.

- استخدم مربع الحوار السابق لحفظ قاعدة البيانات باسم Test.MDB، في الدليل C:\Vb5Prg\Ch16. *بعد إنشاء الملف الجديد يظهر بعدها الشكل 16-5.*

إذا لم تكن ملفات قواعد البيانات والجداول مألوفة لديك، فقط تذكر أن قاعدة البيانات هي عبارة عن مجموعة من البيانات. مثال على ذلك: يمكن لمعمل ما، حفظ جدول بالسلع الموجودة لديه، في قاعدة بيانات.

تعتبر قاعدة البيانات مجموعة من الجداول، مثل جدول أسماء السلع، وجدول عمليات البيع والشراء التي جرت على هذه السلع وهكذا.

ينبغي أن يحتوي الملف Test.MDB جدول واحد، أو أكثر. نرى من الشكل 16-5 قاعدة البيانات Test.MDB لا تحوي أي جدول حتى الآن، ولذلك تصبح مهمتك، إنشاء جدول جديد باسم Parts.



الشكل 16-5

قاعدة البيانات Test.MDB،

لا تحوي أي جدول.

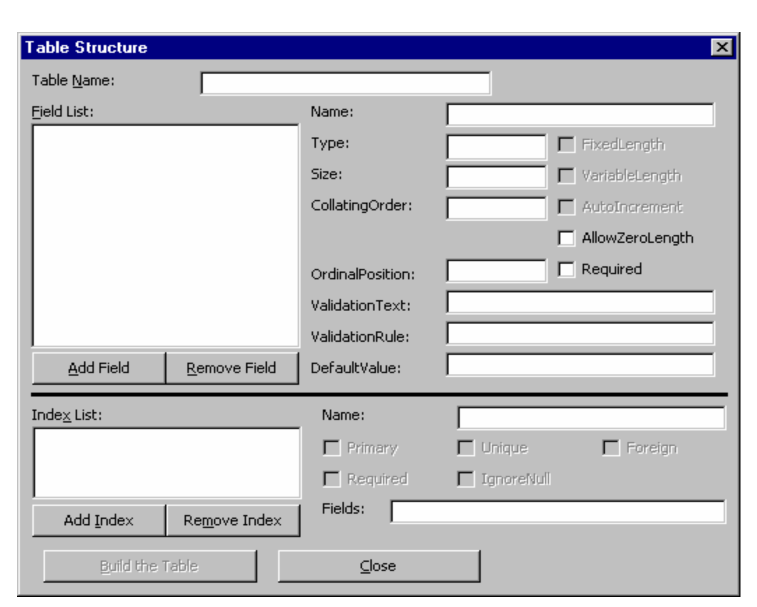
- انقر الزر اليميني للماوس فوق نافذة Database.

ستظهر قائمة فرعية تحوي بندين هما: List Refresh و New Table.

- اختر البند **New Table** لإنشاء جدول جديد.

تظهر النافذة Table Structure كما في الشكل 16-6. استخدم النافذة السابقة لبناء الجدول Parts.

- اكتب Parts في الحقل Table Name، وهو اسم الجدول الجديد.



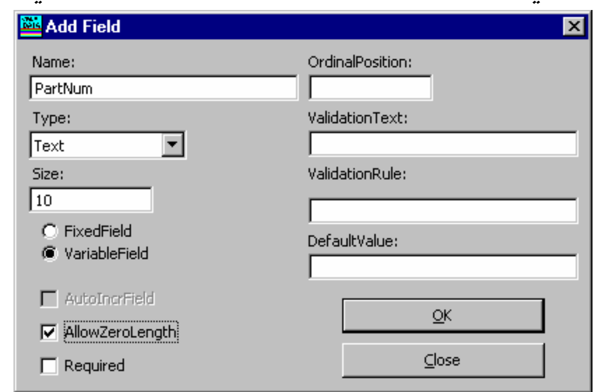
الشكل 16-6

نافذة بناء الجدول.

- أضف إليه الحقول المطلوبة كالتالي:
- انقر الزر Add Field ليظهر مربع الحوار Add Field، كما في الشكل 16-7.

- اكتب PartNum في المربع Name، وهو اسم أول حقل في الجدول Parts.
- حدد النوع Text في المربع Type، لأن الحقل PartNum هوحقل نصي.
- حدد حجم الحقل بعشرة حروف كحد أقصى.

ينبغي أن يكون شكل النافذة Add Field كما في الشكل 16-7.

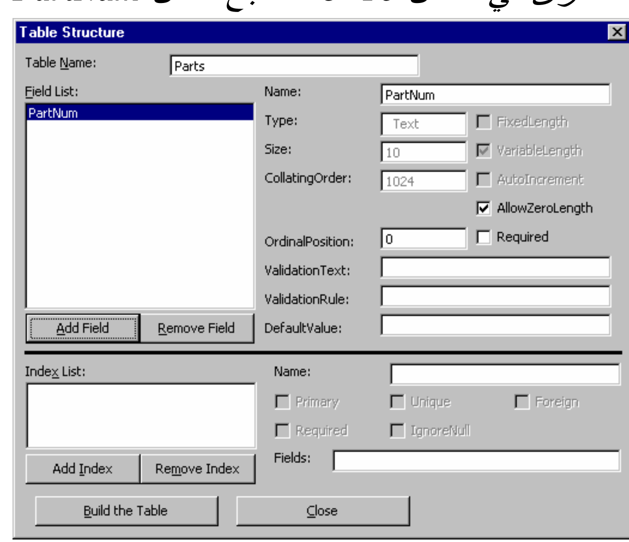


الشكل 16-7

إضافة الحقل الأول.

- انقر الزر **OK** في النافذة Add Field.
- انقر الزر **Close** في النافذة Add Field.

كما ترى في الشكل 16-8. أصبح الحقل PartNum موجوداً في المربع Field List.



الشكل 16-8

نافذة بناء الجدول، بعد

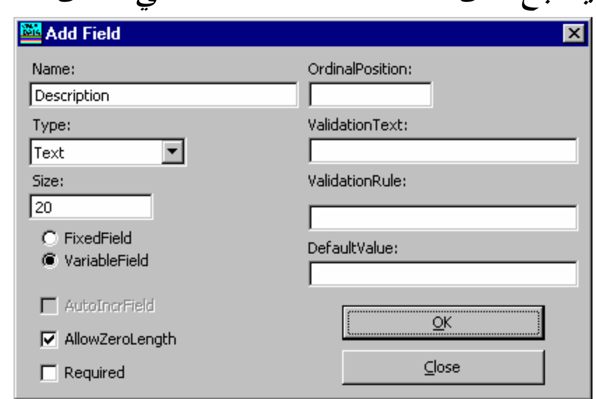
إضافة الحقل PartNum.

- أضف الحقل الثاني للجدول Parts كالتالي:
- انقر الزر **Add Field** في النافذة Table Structure.

تظهر النافذة Add Field.

- اكتب Description في مربع النص Name.
- حدد النوع Text، والحجم 20.

يصبح شكل النافذة Add Field كما في الشكل 16-9.



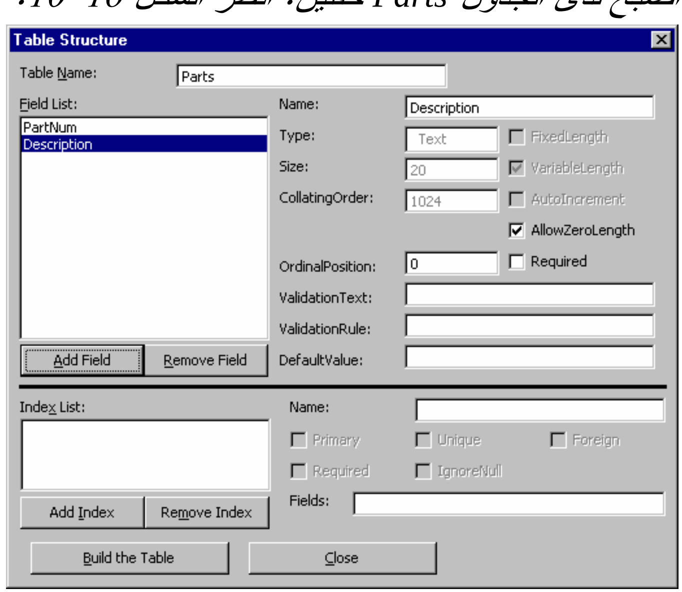
الشكل 16-9

إضافة الحقل

Description.

انقر الزر **OK** ثم الزر **Close** في النافذة Add Field.

أصبح لدى الجدول Parts حقلين. انظر الشكل 16-10.



الشكل 16-10

نافذة بناء الجدول Parts بعد

إضافة حقلين إليه.

لقد انتهيت الآن من تصميم الجدول Parts.

- انقر الزر **Build The Table** في النافذة Table Structure.
- اختر البند **Exit** من القائمة **File**، من نافذة البرنامج VisData.

لقد أنجزت إلى الآن ما يلي:

أنشأت الملف Test.MDB

أنشأت جدولاً يدعى Parts في ملف قاعدة البيانات Test.MDB.

يحتوي الجدول Parts على حقلين هما:

اسم الحقل	النوع	الحجم
PartNum	Text	10
Description	Text	20

إدخال البيانات في الجدول Parts

لإضافة البيانات في الجدول Parts اتبع ما يلي:

- اختر البند **Visual Data Manager** من القائمة **Add Ins**، في فيجول بيسك.

يستجيب فيجول بيسك بإظهار النافذة VisData.

- اختر البند **Open Database** من القائمة **File**، في نافذة البرنامج VisData، ثم اختر البند **Microsoft Access**.

يظهر مربع الحوار Open Microsoft Access Database.

- استخدم مربع الحوار السابق لاختيار الملف Test.MDB التي حفظته في الدليل C:\Vb5Prg\Ch16.

تضم نافذة البرنامج VisData الآن، قاعدة البيانات Test.MDB، انظر الشكل 16-11.

كما ترى، تحوي النافذة Data Bases بندين داخلها هما: Properties و Parts.

لرؤية خصائص قاعدة البيانات، انقر الرمز (+) الظاهر يسار البند Properties.

يمثل البند Parts الجدول Parts المنشأ سابقاً، إذا أضفت المزيد من الجداول لقاعدة البيانات، فإنها ستظهر جميعاً في النافذة Database.



الشكل 16-11

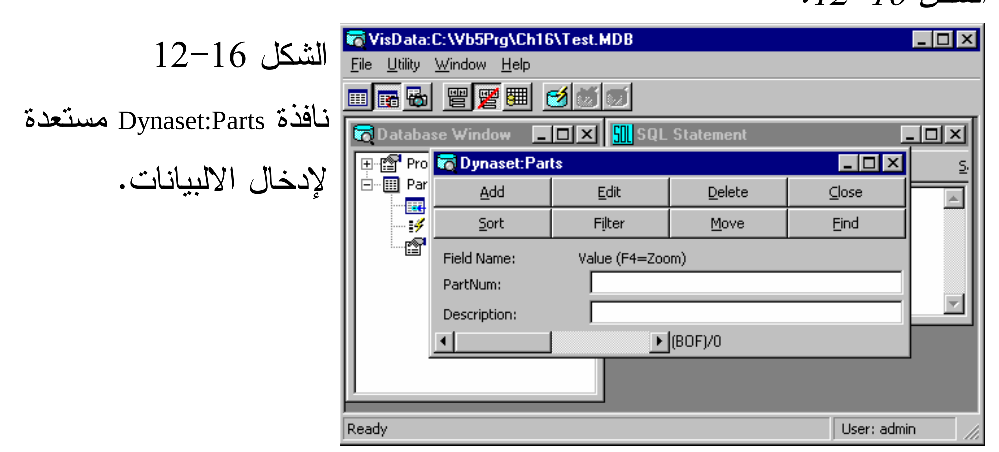
نافذة البرنامج VisData بعد فتح قاعدة البيانات Test.MDB.

- انقر الرمز (+) الظاهر يسار البند Parts. لتظهر لائحة خصائص البند Parts، وأحد هذه البنود هو البند Field.
- انقر الرمز (+) الظاهر يسار البند Field. لتظهر لائحة حقول البند Parts، وهي نفس الحقول التي أنشأتها في الجدول Parts: PartNum و Description.

الآن، أدخل البيانات إلى الجدول Parts كالتالي:

- انقر نقراً مزدوجاً البند Parts في النافذة Database.

تظهر النافذة Dynaset:Parts، يتم إدخال البيانات إلى الجدول، عبر هذه النافذة. انظر الشكل 16-12.

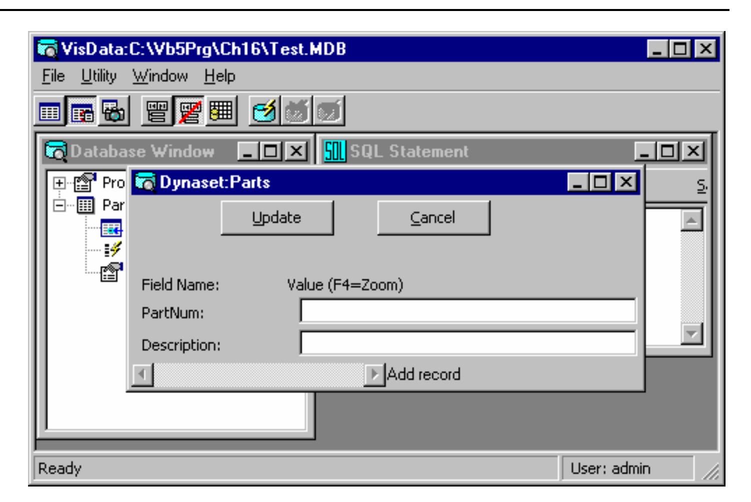


الشكل 16-12

نافذة Dynaset:Parts مستعدة لإدخال الالبيانات.

- انقر الزر **Add** في نافذة Dynaset:Parts، لتتمكن من إضافة سجل جديد للجدول Parts.

يتغير شكل النافذة السابقة، كما في الشكل 16-13.



الشكل 16-13

نافذة Dynaset:Parts بعد نقر الزر **Add**.

- ▫ اكتب PC100 في الحقل PartNum.
- ▫ اكتب PC 100 Megahertz في الحقل Description

ينبغي ظهور النافذة بعد إضافة أول سجل للجدول، كما في الشكل 16-14.

تظهر بيانات السجل الأول في الشكل 16-14. كما قلنا سابقاً، يتألف الجدول Parts من سجلات Records، ويحتوي أول سجل فيه، على القيمة PC100 في الحقل PartNum، والقيمة PC 100 megahertz في الحقل Description.

الشكل 16-14

نافذة Dynaset:Parts بعد كتابة قيم حقلي السجل الأول.

- ▫ انقر الزر **Update** لكتابة السجل الجديد فعلياً إلى الجدول Parts.
- ▫ انقر الزر **Add** لإدخال السجل الثاني، وهو سجل فارغ حالياً.
- ▫ اكتب RAM40 في الحقل PartNum.
- ▫ اكتب RAM 40 nanosecond في الحقل Description.
- ▫ انقر الزر **Update**.

أخيراً، أضف السجل الثالث كالتالي:

- انقر الزر **Add**.
- اكتب Key101 في الحقل PartNum.
- اكتب 101Keys Keyboard في الحقل Description.
- انقر الزر **Update**.
- انقر الزر **Close**.
- اختر البند **Exit** من القائمة **File**.

لقد أنجزت حتى الآن ما يلي:

أنشأت ملف قاعدة بيانات باسم Test.MDB في الدليل C:\Vb5Prg\Ch16.

أنشأت الجدول Parts في قاعدة البيانات.

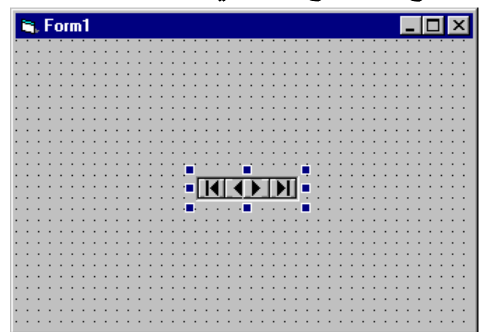
يحتوي الجدول Parts على حقلين هما: PartNum و Description

أضفت ثلاثة سجلات إلى الجدول Parts.

تصميم برنامج البيانات المرئي

- ابدأ بتشغيل فيجول بيسك، وأنشئ مشروعاً من النوع Stardard EXE، واحفظ هذا المشروع في الدليل C:\Vb5Prg\Ch16 باسم Data.Vbp، واحفظ النموذج Form1 باسم Data.frm في نفس الدليل السابق.
- انقر نقراً مزدوجاً على رمز أداة التحكم فى البيانات في نافذة الأدوات.

يصبح النموذج كما في الشكل 16-15.

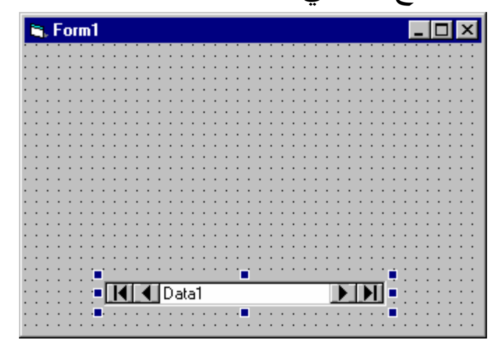


الشكل 16-15

إضافة أداة التحكم فى البيانات إلى النموذج.

تستطيع تكبير أو تصغير أو نقل أداة التحكم فى البيانات مثل باقي عناصر التحكم الأخرى.

- حرّك أداة التحكم فى البيانات إلى أسفل النموذج، ثم كبرها بسحب مقابضها، حتى تصبح كما في الشكل 16-16.



الشكل 16-16

النموذج بعد تغيير موقع وحجم أداة التحكم فى البيانات.

- أسند القيم التالية لخصائص النموذج:

```
Name:          frmData
Caption:       برنامج البيانات
RightToLeft:   True
```

تحديد قاعدة البيانات Access لربطها مع أداة التحكم فى البيانات

ينبغي تحديد اسم ومسار ملف قاعدة البيانات، حتى تتمكن أداة التحكم فى البيانات من الوصول إليها، والتعامل معها. يختلف تحديد قاعدة بيانات، ليست من النوع Access، قليلاً عن تحديد قاعدة بيانات Access.

تعتبر قاعدة البيانات Test.MDB، الذي أنشأتها في بداية هذا الفصل، من النوع Access، وهذا يعني أن الملف متوافق مع البرنامج Microsoft Access.

اتبع الخطوات التالية لتحديد اسم ومسار ملف قاعدة البيانات، المطلوب ربطها مع أداة التحكم فى البيانات Data1:

- تأكد من اختيار أداة التحكم فى البيانات على النموذج، (يظهر حولها ثمانية مقابض)، ثم اختر البند **Properties** من القائمة **View**.

تظهر نافذة الخصائص لأداة التحكم فى البيانات Data1.

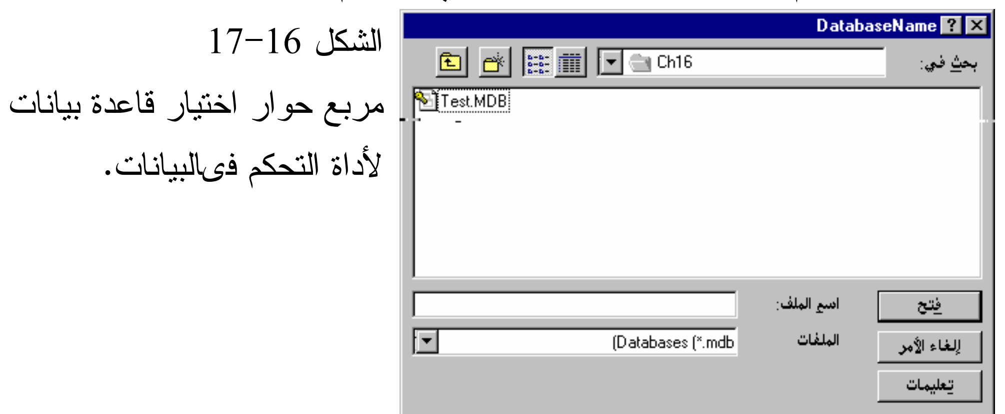
- اختر الخاصية DatabaseName، من نافذة الخصائص، ثم انقر الزر الذي يحوي

ثلاثة نقاط والموجود على نفس سطر الخاصية.

يظهر مربع الحوار DatabaseName، كما في الشكل 16-17.

- استخدم مربع الحوار السابق لاختيار الملف C:\Vb5Prg\Ch16\Test.MDB، المُنشأ سابقاً، ثم انقر الزر **فتح**.

علمت أداة التحكم في البيانات الآن، الملف الذي ستتحكم به، وهو Test.MDB.



الشكل 16-17

مربع حوار اختيار قاعدة بيانات لأداة التحكم فيالبيانات.

- اختر الخاصية RecordSource بنفس الطريقة السابقة، من نافذة الخصائص، ثم انقر الزر الذي يحوي سهماً للأسفل، لسرد لائحة أسماء جميع الجداول الموجودة في ملف قاعدة البيانات الذي اخترته سابقاً.

لو كانت قاعدة البيانات Test.MDB تحوي أكثر من جدول واحد، لظهروا جميعهم في مربع السرد السابق، بما أنها تحوي جدولاً واحداً فقط، لذلك فهو الوحيد الذي يمكن اختياره.

ملاحظة

يجب مراعاة الترتيب في تحديد الخصائص السابقة، بحيث لا يمكن تحديد الخاصية RecordSource قبل تحديد الخاصية DataBaseName.

- اختر اسم الجدول Parts من مربع سرد الخاصية RecordSource.
- أسند القيمة True للخاصية RightToLeft.

وضع مربع نص لعرض البيانات

يجب ربط مربع نص ما، مع أحد حقول جدول قاعدة البيانات Test.MDB، لإظهار محتوياته للمستخدم. لأداء هذه المهمة اتبع ما يلي:

- ضع مربع نص على النموذج frmData، كما في الشكل 16-18. وخصائصه كالتالي:

```
Name:          txtPartNumber
Text:          فراغ
RightToLeft:   True
```

- ضع لافتة على النموذج frmData، كما في الشكل 16-18. وخصائصها كالتالي:

```
Name:          lblPartNumber
Caption:       رقم القطعة
RightToLeft:   True
```



الشكل 16-18

النموذج frmData بعد إضافة مربع نص ولافتة وزر أمر.

- أضف زر أمر إلي النموذج واجعل خصائصه كالتالي:

```
Name:          cmdExit
Caption:       &خروج
RightToLeft:   True
```

ينبغي ظهور النموذج كما في الشكل 16-18.

- اكتب النص التالي للإجراء cmdExit_Click():

```
Private Sub cmdExit_Click()
    End
End Sub
```

يستخدم مربع النص txtPartNumber لإظهار أو تعديل البيانات المسجلة في الحقل PartNum، أخبر فيجول بيسك الآن، أية أداة تحكم فى البيانات تريد ربطها مع مربع النص هذا:

- غيّر الخاصية DataSource لمربع النص txtPartNumber إلى Data1. حيث Data1 هو اسم أداة التحكم فى البيانات الوحيدة الموجودة على النموذج.

بما أن النموذج لديك يحوي فقط أداة تحكم فى البيانات واحدة، لذلك يظهر فقط البند Data1 في مربع سرد الخاصية DataSource.

أخبر فيجول بيسك أيضاً، أن مربع النص هذا، سيُظهر محتويات الحقل PartNum:

- غير الخاصية DataField لمربع النص txtPartNumber إلى القيمة PartNum.
- احفظ عملك الآن، من خلال القائمة **File** ثم البند **Save Project**.

تنبيه

احفظ عملك بصورة متكررة، لأنك لو أخطأت (أو أخطأ فيجول بيسك نفسه)، أثناء تصميم البرنامج، فقد يتسبب ذلك في توقف النظام كاملاً عن العمل، وفقدان كل الأعمال التي لم تحفظها بعد. وظهور الرسالة المألوفة: عليك الاتصال بالبائع، والتي تعني فقدان آخر تعديلاتك. أما إذا حفظت أي تعديل قبل تنفيذ البرنامج مباشرة، ثم حصل خطأ ما، فإنك ببساطة تعيد تنفيذ فيجول بيسك مرة أخرى، وكأن شيئاً لم يكن.

تنفيذ برنامج البيانات

دعنا ننفذ الآن برنامج البيانات لنراه بشكل عملي:

- نفّذ البرنامج بضغط المفتاح F5.

يظهر البرنامج كما في الشكل 16-19. وكما ترى فإن قيمة الحقل التي أدخلتها سابقاً قد ظهرت في مربع النص.



الشكل 16–19
تنفيذ برنامج البيانات.

- انقر أسهم أداة التحكم فى البيانات، ولاحظ كيف تتغير محتويات مربع النص لتُظهر القيمة الفعلية للحقل PartNum.

نقر السهم الداخلي اليساري لأداة التحكم فى البيانات، يُظهر السجل التالي للسجل الحالي، أما نقر السهم الداخلي اليميني، فيُظهر السجل السابق للسجل الحالي، ونقر السهم اليساري الخارجي، يُظهر آخر سجل موجود في الجدول، كما أن نقر السهم اليميني الخارجي، يُظهر أول سجل موجود في الجدول.

- بعد التدرب على برنامج البيانات، انقر الزر **خروج** لإنهاء عمل البرنامج والعودة إلى فيجول بيسك.

كما ترى في الشكل 16–19، تعرض أداة التحكم فى البيانات نصاً داخلها، والنص الافتراضي هو Data1، يمكنك تغير الخاصية Caption لأداة التحكم فى البيانات، لتُظهر عنواناً آخر، وذلك في مرحلة التصميم أو في مرحلة التنفيذ.

فمثلاً يمكن كتابة السطر التالي لتغيير العنوان:

```
Data1.Caption = "Test.MDB قاعدة البيانات"
```

تعطيل مربع النص

لاحظ أن مربع النص الذي يُظهر محتويات الحقلPartNum، هو مربع نص يمكن تغير محتوياته أوتعديلها. جرب المثال التالي:

- نفّذ برنامج البيانات بضغط المفتاح F5.
- غير محتويات الحقل PartNum للسجل الأول، من PC100 إلى PC200.
- انقر السهم الداخلي اليساري لأداة التحكم فى البيانات، لإظهار السجل الثاني.

- انقر السهم الداخلي اليميني لإظهار السجل الأول مرة أخرى.

كما ترى، تغيرت محتويات الحقل PartNum في الجدول.

- انقر الزر **خروج** للعودة إلى فيجول بيسك.
- نفّذ البرنامج مرة أخرى.

لاحظ أن البيانات التي عدلتها سابقاً، قد ظهرت في الحقل PartNum.

- غير محتويات الحقل PartNum إلى PC100 كما كانت سابقاً.
- انقر السهم الداخلي اليساري، ثم السهم الداخلي اليميني، وتأكد أن محتويات الحقل PartNum قد عادت إلى PC100.
- اخرج من البرنامج.

هناك أمور تقنية كثيرة، يجب تضمينها في برنامج البيانات هذا، أو أي برنامج Database احترافي، هي ليست موضوع هذا الفصل.

ببساطة امنع تغير محتويات الحقل PartNum، بتعطيل مربع النص الذي يعرض محتوياته، كالتالي:

- غير قيمة الخاصية Enabled الخاصة بمربع النص من True إلى False.
- احفظ النموذج بضغط المفتاحين Ctrl + S.
- نفّذ البرنامج بضغط المفتاح F5.
- تأكد أن مربع النص يُظهر محتويات الحقل PartNum، بدون إمكانية تعديلها.
- تدرب على برنامج البيانات، ثم اخرج من البرنامج.

تطوير برنامج البيانات

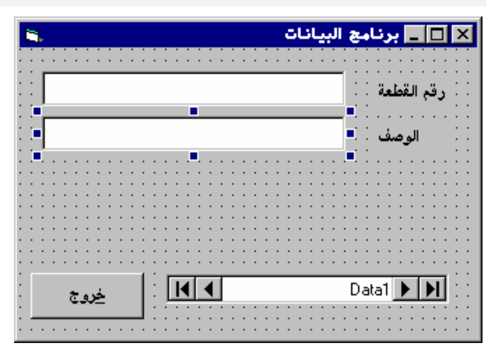
سنطور الآن برنامج البيانات، بإضافة مربع نص آخر إلى النموذج frmData، كما في الشكل 16–20، مهمته إظهار محتويات الحقل Description:

- أضف لافتة Label إلى النموذج، وأسند لها الخصائص التالية:

```
Name:         lblDescription
Caption:      Description
RightToLeft:  True
```

❑ أضف مربع نص إلى النموذج، وأسند له الخصائص التالية:

```
Name:          txtDescription
Text:          فراغ
Enabled:       False
RightToLeft:   True
```



الشكل 16-20

النموذج frmData بعد تطويره.

❑ اربط الآن، مربع النص txtDescription مع أداة التحكم فى البيانات Data1، لتظهر محتويات الحقل Description كالتالي:

❑ أسند لمربع النص txtDescription الخصائص التالية (إضافة للخصائص السابقة):

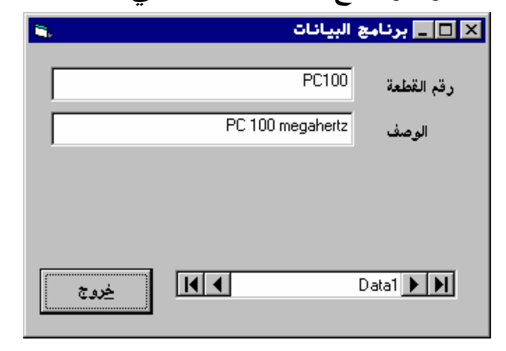
```
DataSource:    Data1
DataField:     Description
```

❑ احفظ النموذج ثانية. (ضغط المفتاحين Ctrl + S).

تنفيذ برنامج البيانات المطور

❑ نفّذ البرنامج الآن بضغط F5:

يظهر برنامج البيانات، كما في الشكل 16-21.



الشكل 16-21

برنامج البيانات بعد تطويره.

- انقر أسهم أداة التحكم فى البيانات، للانتقال بين السجلات.

كما تلاحظ، تمثل محتويات مربعي النص، محتويات السجل الحالي.

- اخرج من البرنامج.

إضافة حقل منطقي Logical إلى جدول قاعدة البيانات

بطريقة مشابهة، يمكن إضافة حقل منطقي Logical إلى الجدول Parts. فمثلاً يمكن إضافة الحقل InStock، الذي يحوي قيمة منطقية هي: True أو False. عندما تكون قيمة الحقل InStock هى True، فهذا يعني تواجد المادة ذات الرقم PartNumber في المخزن. أما إذا كانت قيمة الحقل InStock القيمةFalse، فهذا يعني عدم تواجد المادة ذات الرقم PartNumber في المخزن.

يمكن وضع خانة اختيار CheckBox، على النموذج، وربطها مع الحقل InStock الموجود في الجدول Parts، وذلك كالتالي:

- أضف خانة اختيار إلى النموذج frmData. وأسند لها الخصائص التالية:

```
DataSource:   Data1
DataField:    InStock
RightToLeft:  True
```

استخدام عناصر التحكم المُلْزَمَة Bound Controls

وضح برنامج البيانات كيفية إلزام مربع النص بإظهار محتويات الحقل الموجود في الجدول. تُسمى الأدوات التي تُظهِر محتويات حقول جداول قواعد البيانات، بالعناصر الملزمة Bound Controls.

يمكن استخدام مربعات النصوص، وخانات الاختيار، في البرنامج، كأدوات مُلزَمَة. كما يمكن إضافة حقل صورة، إلى الجدول Parts، يخزن صوراً Bmp، وإضافة عنصر الصورة إلى النموذج، وربطه مع حقل الصورة، وسيعمل عنصر الصورة على إظهار الصورة المخزنة في الحقل.

خصائص وطرق Methods أداة التحكم فى البيانات

يوجد لأداة التحكم فى البيانات خصائص عديدة، وطرق مختلفة لمعالجة البيانات، والتحكم بها، وسيشرح هذا الفصل بعضاً منها:

الطريقة Refresh (إنعاش)

يمكن استخدام الطريقة Refresh، لتحديث البيانات المرتبطة مع أداة التحكم فى البيانات، ومعرفة آخر التعديلات التي جرت على ملف قاعدة البيانات. مثال، تصور وجود ملف قاعدة البيانات في كومبيوتر رئيسي، وموصول مع جهازك الشخصي من خلال شبكة محلية LAN (Local Area Network) ، ووجود مستخدم آخر، يعدّل نفس ملف قاعدة البيانات. قد لا تتطابق البيانات الموجودة على شاشتك، مع نفس البيانات التي عدّلها المستخدم الآخر.

لضمان الحصول على آخر التعديلات التي جرت من قبل المستخدمين الآخرين، يجب استخدام الطريقة Refresh على أداة التحكم فى البيانات، لضمان إظهار البيانات الصحيحة على شاشتك، وأنها هي نفسها، آخر ما جرى من تعديلات على ملف قاعدة البيانات.

الخاصية Exclusive (وصول وحيد)

إذا أردت أن يكون برنامجك هو الوحيد القادر على الوصول إلى ملف قاعدة البيانات، وبدون السماح لأي مستخدم آخر بالوصول أيضاً، عليك استخدام الخاصية Exclusive وتغيرها إلى True وذلك كالتالي:

```
Data1.Exclusive = True
```

تمنع بهذه الطريقة، أي برنامج أو مستخدم آخر من قدرته على فتح نفس ملف البيانات.

القيمة الافتراضية لهذه الخاصية هي False، ولإعادتها إلى الوضع الافتراضي (السماح للمستخدمين الآخرين بالوصول أيضاً)، اكتب السطر التالي:

```
Data1.Exclusive = False
```

أما إذا كانت قاعدة البيانات مفتوحة مسبقاً، وغيرت هذه الخاصية، فيجب تنفيذ الطريقة Refresh بعدها مباشرة، لتنفيذ الأمر بشكل صحيح، وذلك كالتالي:

```
Data1.Exclusive = False
Data1.Refresh
```

يعتبر الوصول إلى قاعدة بيانات مفتوحة بشكل خاص (Exclusive = True)، أسرع بكثير من الوصول إليها وهي مفتوحة للعمل المشترك (Exclusive = False). ولكن لا تلجأ لهذا الأسلوب، إلا إذا كنت متأكداً أنك الوحيد الذي يعمل على هذا الملف، وإلا، فإن المستخدمين الآخرين لن يستطيعوا العمل أيضاً، طالما كنت مستمراً في عملك.

الخاصية ReadOnly (للقراءة فقط)

لقد رأيت سابقاً، أن تغيير محتويات مربع النص، والمرتبط مع حقل أحد جداول قاعدة البيانات، يغير قيمة هذا الحقل فعلياً، بمجرد الانتقال لسجل آخر.

ماذا لو أردت استعراض محتويات قاعدة البيانات، دون الحاجة لتعديلها، أو أردت طريقة عملية، لمنع التعديل الحاصل نتيجة خطأ غير مقصود من أحد المستخدمين.

إذا أردت فتح ملف البيانات للقراءة فقط، غير الخاصية ReadOnly إلى True، وذلك من خلال الإجراء Form_Load(). اكتب الأسطر التالية:

```
Prvat Sub Form_Load()
    Data1.ReadOnly = True
    Data1.Refresh
End Sub
```

- فعّل مربع النص (Enabled = True) الذي يُظهر محتويات الحقل PartNum.
- احفظ المشروع.
- نفّذ البرنامج.
- غير محتويات مربع النص للسجل الأول.

عند هذه النقطة، ستتغير محتويات مربع النص فقط، بدون تغير محتويات الحقل الفعلي، كما سترى في الخطوات اللاحقة.

- انتقل للسجل الثاني بنقر الزر اليساري الداخلي لأداة التحكم فى البيانات.

- انتقل للسجل الأول بنقر الزر الداخلي اليميني.

لاحظ عدم تغير قيمة الحقل الأصلية، لأن الخاصية ReadOnly لها القيمة True.

- لإعادة قيمة الخاصية ReadOnly إلى القيمة False، عدّل محتويات الإجراء Form_Load() كالتالي:

```
Prvat Sub Form_Load()
    Data1.ReadOnly = False
    Data1.Refresh
End Sub
```

- جرب الخطوات السابقة مرة أخرى، ولاحظ أن التعديلات التي أجريتها في مربع النص، قد ثُبتت بالفعل في ملف قاعدة البيانات.

للمرة الثانية نقول، يجب استخدام الطريقة Refresh، بعد كل عملية تغيير أحد خصائص أداة التحكم فى البيانات.

ملاحظة

تأكد من إعادة قيمة الخاصية ReadOnly إلى False مرة أخرى، لضرورة إكمال بقية التمارين في هذا الفصل.

عند تصميم برنامج قاعدة بيانات احترافي، ينبغي إخبار المستخدم (بطريقة ما)، أن قاعدة البيانات هي للقراءة فقط (إذا كانت كذلك)، وإلا فإنه سيعدّل محتويات مربعات النصوص، ظاناً أن تعديلاته تُخزن فعلياً في ملف قاعدة البيانات، وهي لا تخزن عملياً.

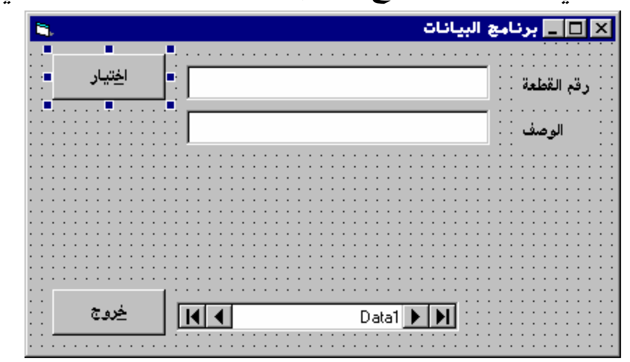
استخدام عبارات لغة الاستعلام البنيوية SQL

يمكن استخدام عبارات SQL، لاختيار وتحديد مجموعة من السجلات التي تحقق شرطاً (أو شروطاً) معينة. اتبع الخطوات التالية لعمل ذلك:

- ضع زر أمر على النموذج frmData، وأسند له الخصائص التالية:

```
Name:          cmdSelect
Caption:       ا&ختيار
RightToLeft:   True
```

*ينبغي ظهور النموذج بعد إضافة الزر **اختيار** كما في الشكل 16-22.*



الشكل 16-22

النموذج frmData

بعد إضافة الزر **اختيار**.

- أضف الأسطر التالية إلى الإجراء cmdSelect_Click():

```
Private Sub cmdSelect_Click ()
    Data1.RecordSource = "SELECT * FROM Parts _
                        & "WHERE PartNum = 'PC100'"
    Data1. Refresh
End Sub
```

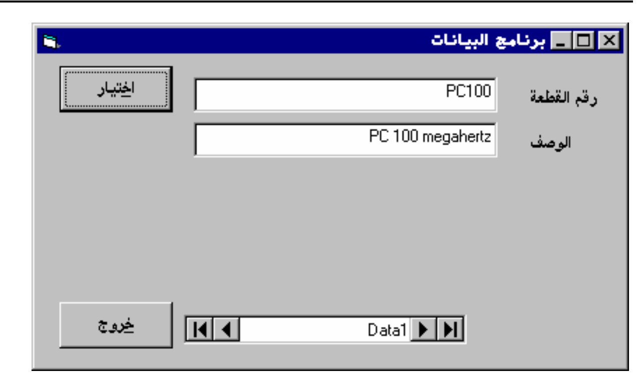
تُكتب عبارات SQL عادة، بأحرف كبيرة، لتمييزها عن باقي العبارات الأخرى. يختار الإجراء السابق عند تنفيذه، كل السجلات الموجودة في الجدول Parts، والموافقة لشرط مساواة قيمة الحقل PartNum للقيمة PC100. ثم تنفذ الطريقة Refresh عبارة SQL فعلياً.

- احفظ عملك ونفذ البرنامج.
- تأكد أن قيمة PartNum للسجل الأول تساوي PC100.
- استخدم أسهم أداة التحكم في البيانات لاستعراض السجلات المختلفة.

كما ترى، تستطيع التنقل بين كل السجلات.

- انقر الزر **اختيار**.

بسبب احتواء قاعدة البيانات سجلاً واحداً فقط، مطابقاً للشرط الذي وضعناه في عبارة SQL، فسيظهر هذا السجل لوحده فقط، ويصبح البرنامج كما في الشكل 16-23.



الشكل 16-23
نقر الزر **اختيار** يؤدي لتحديد سجل واحد فقط.

لن يتغير السجل الظاهر، بنقر أسهم أداة التحكم فى البيانات، بسبب عدم وجود غيره، والمطابق للشرط السابق. تقوم عبارة SQL بتصفية كل السجلات التي لا توافق الشرط المحدد، وتعمل على استبعادها.

- اخرج من البرنامج.

يمكن استخدام عبارة SQL لاختيار أية مجموعة من السجلات. مثلاً، افترض أنك أضفت الحقل InSock إلى الجدول Parts (يدلك هذا الحقل على وجود المادة في المخزن)، وأردت معرفة جميع المواد الموجودة في مخزنك، سيكون شكل عبارة SQL مشابهاً لما يلي:

```
Private Sub cmdSelect_Click ()
    Data1.RecordSource = "SELECT * FROM Parts _
                         & "WHERE InSock = True"
    Data1.Refresh
End Sub
```

عند تنفيذ البرنامج، ثم نقر الزر **اختيار**، فسوف تظهر في برنامج البيانات، كل السجلات التي توافق المواد المتوفرة في المخزن (InStock = True).

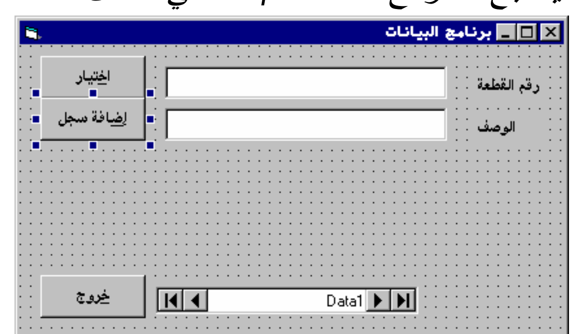
الطريقة AddNew (إضافة سجل)

تستخدم الطريقة AddNew لإضافة سجل جديد في مرحلة التنفيذ:

- أضف زر أمر للنموذج frmData، وأسند له الخصائص التالية:

```
Name:          cmdAddRecord
Caption:       إ&ضافة سجل
RightToLeft:   True
```

ليصبح النموذج frmData كما في الشكل 16-24.



الشكل 16-24

النموذج frmData بعد إضافة الزر **إضافة سجل**.

- أضف الأسطر التالية للإجراء cmdAddRecord_Click():

```
Private Sub cmdAddRecord_Click()
    Data1.RecordSet.AddNew
End Sub
```

تُطبق الطريقة AddNew على الخاصية Recordset التابعة لأداة التحكم فى البيانات. (ستشرح الخاصية Recordset لاحقاً).

في التمرين التالي، سنضيف بعض السجلات للجدول Parts:

- غير الخاصية Enabled لمربعي النص، إلى القيمة True.
- احفظ المشروع ونفذ البرنامج.
- انقر الزر **إضافة سجل**.

*بمجرد نقر الزر **إضافة سجل**، يُظهر برنامج البيانات سجلاً خالياً من البيانات، بمعنى آخر، تمسح كل الأدوات المرتبطة مع أداة التحكم فى البيانات، محتوياتها، استعداداً لكتابة بيانات جديدة عليها.*

- املأ محتويات مربعي النص بالبيانات الجديدة.

لتحفظ السجل الجديد في ملف قاعدة البيانات، عليك الانتقال إلى سجل آخر، لذلك انقر السهم الداخلي اليميني، للانتقال إلى السجل السابق.

- اخرج من البرنامج.

الطريقة Delete (حذف سجل)

لحذف السجل الحالي من قاعدة البيانات، استخدم الطريقة Delete:

- أضف زر أمر للنموذج frmData، وضعه أسفل الزر السابق، ثم أسند له الخصائص التالية:

```
Name:          cmdDelete
Caption:       &حذف
RightToLeft:   True
```

- أضف الأسطر التالية للإجراء cmdDelete_Click():

```
Private Sub cmdDelete_Click()
    Data1.RecordSet.Delete
    Data1.RecordSet.MoveNext
End Sub
```

- احفظ المشروع ثم نفّذ البرنامج.
- استخدم الأسهم للانتقال إلى السجل الذي ترغب بحذفه.
- انقر الزر **حذف** لحذف السجل الحالي الظاهر.
- تدرب على الزرين **إضافة سجل** و **حذف** لفترة، ثم اخرج من البرنامج.

ملاحظة

تعتبر عملية حذف سجل ما، أمراً بالغ السهولة، وما عليك سوى نقر الزر **حذف**. هذه السهولة الشديدة غير مقبولة مطلقاً في الحياة العملية، أو البرامج التجارية. فأنت تحتاج إلى تحذير المستخدم من عملية الحذف، وإظهار مربع حوار يسأله إذا كان متأكداً من حذف السجل، ولا يُحذف السجل فعلياً، إلا في حال موافقة المستخدم على ذلك.

الطريقة MoveNext (سجل لاحق)

تُظهر هذه الطريقة السجل التالي للسجل الحالي، وتقوم بنفس وظيفة السهم الداخلي اليساري، ولها الشكل التالي:

```
Data1.Recordset.MoveNext
```

الطريقة MovePrevious (سجل سابق)

تُظهر هذه الطريقة السجل السابق للسجل الحالي، وتقوم بنفس وظيفة السهم الداخلي اليميني، ولها الشكل التالي:

```
Data1.Recordset.MovePrevious
```

الطريقة MoveLast (سجل أخير)

تُظهر هذه الطريقة آخر سجل موجود في مجموعة السجلات، وتقوم بنفس وظيفة السهم الخارجي اليساري، ولها الشكل التالي:

```
Data1.Recordset.MoveLast
```

الطريقة MoveFirst (سجل أول)

تُظهر هذه الطريقة أول سجل موجود في مجموعة السجلات، وتقوم بنفس وظيفة السهم الخارجي اليميني، ولها الشكل التالي:

```
Data1.Recordset.MoveFirst
```

الخاصية Recordset (مجموعة السجلات)

تعتبر هذه الخاصية، الكائن الذي يحوي مجموعة من السجلات، قد تكون هذه السجلات موجودة في جدول معين (مثل الجدول Parts)، أو قد تكون مجموعة من السجلات التي تحقق شرطاً معيناً (PartNum = PC100)، أو قد تكون مجموعة من السجلات الموجودة في أكثر من جدول.

مثال، إذا لم تقم بفرز أو تصفية للسجلات، بواسطة عبارة SQL، فإن هذه الخاصة تمثل كل السجلات الموجودة في الجدول Parts.

يمكن لأداة التحكم فى البيانات التعامل مع أكثر من جدول واحد، لذلك يمكن إنشاء مجموعة سجلات RecordSet، هي عبارة عن سجلات أكثر من جدول واحد.

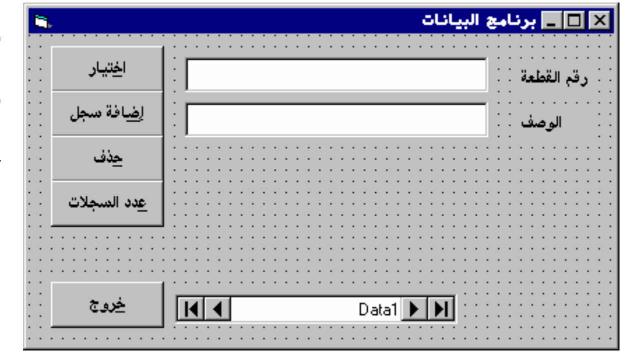
توضح لك الخطوات التالية كيفية معرفة عدد السجلات الموجودة في مجموعة السجلات الحالية:

- أضف زر أمر للنموذج frmData، وأسند له الخصائص التالية:

```
Name:           cmdCountRecords
```

```
Caption:        &عدد الـسجلات
RightToLeft:    True
```

- بعد إضافة الزر **عدد السجلات**، يصبح النموذج كما في الشكل 16-25.



الشكل 16-25

النموذج frmData بعد إضافة الزر **عدد السجلات**.

- أضف الأسطر التالية للإجراء ()cmdCountRecords_Click:

```
Privat Sub cmdCountRecords_Click()
    Data1.Recordset.MoveLast
    MsgBox Data1.RecordSet.RecordCount
End Sub
```

ينقل الإجراء السابق المؤشر، إلى آخر سجل موجود في مجموعة السجلات الحالية، وتحصي الطريقة RecordCount عدد السجلات، من أول سجل وحتى السجل الحالي (آخر سجل)، ولذلك فهي تعود بقيمة عدد السجلات الفعلي الموجودة في الجدول Parts. ثم تُظهر مربع رسالة، لتخبر المستخدم عن عدد السجلات الفعلي.

- احفظ المشروع ثم نفّذ البرنامج.
- انقر الزر **عدد السجلات**.

تظهر رسالة تُخبر المستخدم عن عدد السجلات الصحيح.

- أضف بعض السجلات للجدول Parts، ثم انقر الزر **عدد السجلات**، لتعرف عدد السجلات الحالي.
- اخرج من البرنامج.

الخاصية Value

يمكن معرفة قيمة حقل ما، بقراءة الخاصية Text، التي تعطي محتويات النص الموجود ضمن مربع نص. فمثلاً، يمكن معرفة قيمة الحقل PartNum للسجل الحالي، بقراءة الخاصية Text التابعة لمربع النص txtPartNumber.

- أضف السطر التالي للإجراء Form_Click()، وهو ينفذ عند نقر سطح النموذج مرة واحدة:

```
Private Sub Form_Click()
    MsgBox "PartNumber:" + txtPartNumber.Text
End Sub
```

- احفظ المشروع ومن ثم نفذه.
- انقر سطح النموذج مباشرة (المساحة الخالية منه).

تظهر رسالة، تخبرك عن قيمة الحقل PartNum.

- اخرج من البرنامج.

تحتاج أحياناً، لمعرفة قيمة حقل ما، غير مرتبط مع مربع نص، أو أي عنصر آخر، لذلك سنستخدم الخاصية Value لمعرفة قيمة الحقل مباشرة، اتبع ما يلي:

- عدّل الأسطر الموجودة في الإجراء Form_Click() إلى ما يلي:

```
Private Sub Form_Click()
    Dim MyString As String

    MyString = Data1.Recordset.Fields("PartNum").Value
    MsgBox MyString
End Sub
```

- احفظ المشروع ومن ثم نفّذ البرنامج.
- انقر سطح النموذج.

تظهر رسالة تخبرك عن قيمة الحقل PartNum للسجل الحالي.

- اخرج من البرنامج.

يقوم الإجراء السابق، بوضع قيمة الحقل PartNum في المتحول MyString، ثم يُظهر المتحول من خلال العبارة MsgBox.

يمكنك بهذه الطريقة معرفة قيمة أي حقل من حقول السجل الحالي، ولو كان هذا الحقل غير مرتبط مع أي عنصر تحكم.

الخاصيتين EOF و BOF

يمكن استخدام الخاصية EOF (End Of File) والخاصية BOF (Beginning Of File)، لمعرفة أن السجل الحالي، سجل صحيح وموجود. مثال، لو أظهر برنامجك سجل محدد، وقام أحد المستخدمين الآخرين على الشبكة، بحذف هذا السجل الذي تعرضه على شاشتك، هذا سيعني أن مؤشر السجل لديك، أصبح يشير إلى سجل غير صحيح أو غير موجود. يمكن لبرنامجك معرفة أن مؤشر السجل يشير إلى سجل صحيح أو موجود كالتالي:

```
If Data1.Recordset.EOF = False _
            And Data1.Recordset.BOF = False Then
'''''''''''''''''''''''''''''''''
'''' مؤشر السجل يشير لسجل صحيح''''
'''''''''''''''''''''''''''''''''
End If
```

تفحص كتلة If .. End If السابقة، إذا كان مؤشر السجل، يشير لسجل صحيح أو موجود، وذلك بفحص كلتا الخاصيتين BOF و EOF، فإذا كانت قيمتهما مساوية للقيمة False، يعني أن مؤشر السجل يشير لسجل صحيح وموجود.

أي اختلاف في أحد هاتين القيمتين عن False، يعني أن مؤشر السجل يشير إلى سجل خاطئ أو غير موجود (تم حذفه من قبل مستخدم آخر).

جدول 16-1 احتمالات قيم الخاصيتين BOF و EOF:

إذا كانت قيمة BOF	وكانت قيمة EOF	يكون:
False	False	المؤشر يشير إلى سجل حقيقي.
False	True	المؤشر يشير إلى ما بعد آخر سجل في الجدول.

المؤشر يشير إلى ما قبل أول سجل في الجدول.	False	True
لا يحتوي الجدول على أية سجلات.	True	True

ماذا يمكن لأداة التحكم فى البيانات فعله أيضاً ؟

قدمنا لك في هذا الفصل، بعض الخصائص والطرق الأولية لأداة التحكم فى البيانات. يمكن الاستفادة من هذه الخصائص والطرق، في إظهار أو تعديل محتويات حقول جداول قاعدة البيانات.

تدعم أداة التحكم فى البيانات العديد من المزايا، التي تسمح لفيجول بيسك بمعالجة قواعد البيانات والتحكم بها. مثلاً، يمكنك بناء ملف قاعدة بيانات أو تعديل البنية التركيبية له. في الحقيقة، فإن أداة التحكم فى البيانات هي أقوى بكثير مما تتوقع، وسهولتها النسبية لا تلغي قوتها أبداً.

تستطيع أداة التحكم فى البيانات معالجة أنواع أخرى من قواعد البيانات المختلفة عن Microsoft Access، وهي تستخدم نفس مكتبة محرك البيانات DBEngine الذي يستخدمه البرنامج الشهير Access، يضمن لك هذا الأمر، المصداقية والثقه والسرعة وخلو ملفات قواعد البيانات التي تستخدمها من الأخطاء، وتوافقها مع برامج قواعد البيانات الأخرى.

تعتبر قواعد البيانات من النوع Access، من أسهل وأقوى أنواع قواعد البيانات، في نفس الوقت.

الخلاصة

تعرفت في هذا الفصل، على أداة التحكم فى البيانات الموجودة مع لغة فيجول بيسك نفسها، وكيفية تغيير بعض خصائصها المهمة، وربطها مع عناصر أخرى، مثل مربعات النص، لإظهار محتويات الحقول الموجودة في الجدول.

تعلمت أيضاً كيفية إنعاشها، لمعرفة أخر المستجدات. وتعلمت أيضاً مبادىء عبارات SQL، وكيفية إضافة أو حذف أو إستعراض السجلات الموجودة فى الجدول.

الفصل الثامن عشر

محاكاة ضربات المفاتيح

سوف تتعلم في هذا الفصل، كيفية إرسال ضربات المفاتيح عبر برنامجك إلى برامج ويندوز الأخرى، وكيفية توجيه ضربات المفاتيح من البرنامج الذي أنشأته إلى البرنامج نفسه مرة أخرى، تفيد هذه التقنية في إنشاء برامج دعائية Demo أو أية برامج أخرى.

محاكاة ضربات المفاتيح: برنامجي المصدر والمقصد

تستطيع كتابة برنامج ما، عبر لغة فيجول بيسك، يُرسل ضربات المفاتيح، وكأنها أرسلت من لوحة المفاتيح Keyboard الفعلية. سنكتب الآن برنامجين، Source.exe و Dest.exe. يولد البرنامج Source.exe ضربات المفاتيح، ويستقبل البرنامج Dest.exe هذه الضربات المنشأة من البرنامج Source.exe.

التصميم المرئي لبرنامج المصدر

- ابدأ مشروعاً جديداً من النوع Standard EXE، واحفظ المشروع باسم Source.Vbp في الدليل C:\VB5Prg\Ch18، واحفظ النموذج باسم Source.frm في نفس الدليل.
- أنشأ النموذج frmSource وفقاً للجدول 18-1.

جدول 18-1. جدول خصائص النموذج frmSource.

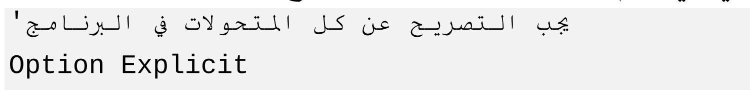
الكائن	الخاصية	القيمة
Form	**Name**	**frmSource**
	Caption	برنامج المصدر
	RightToLeft	True
CommandButton	**Name**	**cmdExit**

	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdSend**
	Caption	إر&سال
	RightToLeft	True
TextBox	**Name**	**txtUserArea**
	MultiLine	True
	ScrollBars	3-Both
	RightToLeft	True

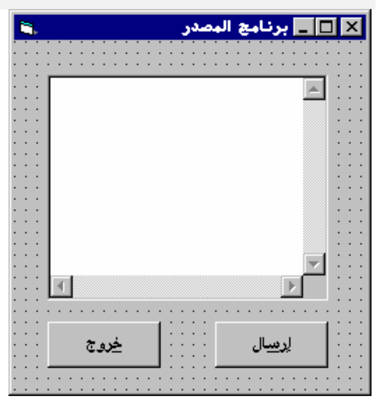
بعد انتهاء النموذج ينبغي أن يصبح كما في الشكل 18-1.

إدخال نص البرنامج frmSource

□أدخل النص التالي في قسم التصريحات العامة للنموذج frmSource:



```
'يجب التصريح عن كل المتحولات في البرنامج
Option Explicit
```



الشكل 18-1

النموذج frmSource

بعد انتهاء تصميمه.

□أدخل النص التالي في الإجراء ()Form_Load:

```
Private Sub Form_Load()
    Dim ID
    'تنفيذ برنامج المقصد
    ChDir App.Path
    ID = Shell("Dest.exe", vbNormalFocus)
End Sub
```

□أدخل النص التالي في الإجراء cmdSend_Click():

```
Private Sub cmdSend_Click()
    'جعل برنامج المقصد فعالا
    AppActivate "برنامج المقصد"

    'إرسال الأحرف إلى برنامج المقصد
    SendKeys txtUserArea.Text, True
End Sub
```

□أدخل النص التالي في الإجراء cmdExit_Click():

```
Private Sub cmdExit_Click()
    End
End Sub
```

□احفظ المشروع.

□أنشئ الملف التنفيذي Source.exe، باختيار البند **Make Source.exe...** من القائمة **File**، واحفظه في الدليل C:\VB5Prg\EXE.

لا تستطيع تنفيذ البرنامج Source.exe بعدْ، لأنه يحتاج إلى البرنامج Dest.exe لكي يعمل بصورة طبيعية.

التصميم المرئي لبرنامج المقصد

□ابدأ مشروعاً جديداً من النوع Standard EXE، واحفظ المشروع في الدليل C:\VB5Prg\Ch18 باسم Dest.Vbp، واحفظ النموذج الجديد باسم Dest.Frm في نفس الدليل.

□أنشأ النموذج frmDest وفقاً للجدول 18-2.

جدول 18-2. جدول خصائص النموذج frmDest.

الكائن	الخاصية	القيمة
Form	**Name**	**frmDest**
	Caption	برنامج المقصد
	RightToLeft	True
CommandButton	**Name**	**cmdExit**

	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdDisplayMessage**
	Caption	&عرض الرسالة
	RightToLeft	True

بعد انتهاء النموذج ينبغي أن يصبح كما في الشكل 18-2.



الشكل 18-2

النموذج frmDest

بعد انتهاء تصميمه.

إدخال نص برنامج المقصد

□أدخل النص التالي في قسم التصريحات العامة للنموذج frmDest:

```
'يجب التصريح عن كل المتحولات في البرنامج
Option Explicit
```

□أدخل النص التالي في الإجراء ()cmdDisplayMessage_Click:

```
Private Sub cmdDisplayMessage_Click()
    MsgBox "لقد نقرت زر عرض الرسالة"
End Sub
```

□أدخل النص التالي في الإجراء ()cmdExit_Click:

```
Private Sub cmdExit_Click()
    End
End Sub
```

سوف ننشئ الآن الملف التنفيذي Dest.exe:

□اختر البند **...Make Dest.exe** من القائمة **File**، واحفظ الملف التنفيذي Dest.exe في الدليل C:\VB5Prg\EXE.

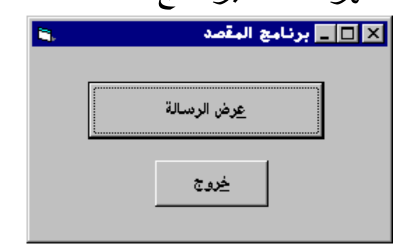
□تأكد من وجود الملفين التنفيذيين Source.exe و Dest.exe في نفس الدليل السابق.

تنفيذ برنامجي المصدر والمقصد معاً

نفّذ أولاً البرنامج Dest.exe:

- ▫أنهِ فيجول بيسك، من القائمة **File**، البند **Exit**
- ▫افتح الدليل C:\VB5Prg\EXE من المستكشف، ثم نفّذ البرنامج Dest.exe، بالنقر المزدوج على رمزه، كأي برنامج آخر.

تظهر نافذة البرنامج Dest.exe كما في الشكل 18-3.



الشكل 18-3

نافذة برنامج المقصد.

- ▫انقر الزر **عرض الرسالة** في البرنامج Dest.exe.

تظهر الرسالة التالية "لقد نقرت زر عرض الرسالة " كما في الشكل 18-4.



الشكل 18-4

الرسالة التي تظهر عقب نقر الزر **عرض الرسالة**.

- ▫انقر الزر **خروج** في البرنامج Dest.exe لإنهائه.

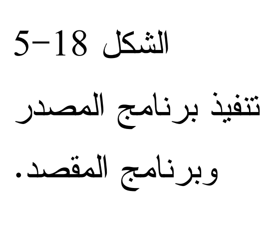
لاحظ بساطة برنامج المقصد، واحتوائه على زرين فقط، أحدهما لإظهار رسالة، والثاني للخروج.

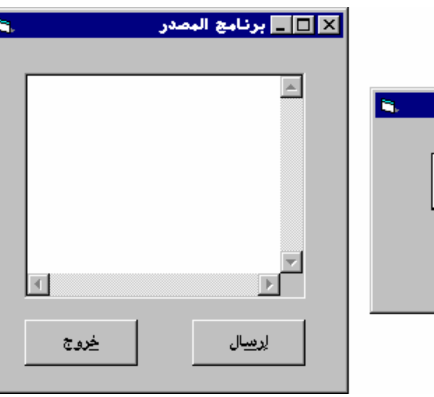
الهدف من البرنامجين السابقين، توضيح فكرة استطاعتك نقر الزر **عرض الرسالة** الموجود في برنامج المقصد، من خلال برنامج المصدر.

اتبع الخطوات التالية لرؤية ذلك فعلياً:

- ▫تأكد أن البرنامج Dest.exe غير منفذ حالياً.
- ▫نفّذ البرنامج Source.exe.

يتم تنفيذ البرنامج Source.exe، وآلياً سوف يتم تنفيذ البرنامج Dest.exe كما في الشكل 18-5.





الشكل 18-5
تنفيذ برنامج المصدر وبرنامج المقصد.

صغّر كافة النوافذ ماعدا نافذتي البرنامجين السابقين، وضعهما بشكل متجاور، بحيث تراهما بشكل واضح ومنفصل.

- اكتب %ع (التي تعني ضغط المفتاحين Alt+ع في لوحة المفاتيح) في مربع النص الموجود في برنامج المصدر.
- انقر الزر **إرسال** في برنامج المصدر، لإرسال المفتاحين السابقين إلى برنامج المقصد.

كما ترى، أظهر برنامج المقصد الرسالة الخاصة به، وكأن أحدهم قد نقر الزر **عرض الرسالة**، (لم يفعل أحد ذلك)، وإنما قلّد البرنامج Source.exe إرسال ضربات المفاتيح، وكأنها أرسلت من لوحة المفاتيح فعلياً.

- اكتب هذه المرة %خ (التي تعني ضغط المفتاحين Alt+خ في لوحة المفاتيح) في مربع النص الموجود في برنامج المصدر (تذكر مسح محتويات مربع النص أولاً).
- انقر الزر **إرسال** في برنامج المصدر، لإرسال المفتاحين السابقين إلى برنامج المقصد.

ماذا حصل، لقد تم إنهاء برنامج المقصد، عن طريق نقر الزر **خروج**، وأيضاً لم يفعل أحد ذلك، وإنما تم هذا الأمر عن طريق برنامج المصدر، الذي أرسل ضربتي المفتاحين Alt+خ إلى برنامج المقصد، وبدوره أُجبر على الخروج.

- جرب كتابة % (إضافة لحرف مسافة واحد بعده)، التي تعني ضغط المفتاحين Alt+Space في لوحة المفاتيح.

▫انقر الزر **إرسال** في برنامج المصدر، لإرسال المفتاحين السابقين إلى برنامج المقصد.

لقد ظهرت قائمة النظام System Menu، في برنامج المقصد.

▫أضف للكتابة السابقة (%حرف مسافة واحد)، الحرف (ك)، لتصبح (% ك)، ثم أعد نقر الزر **إرسال**.

لقد أخذت نافذة البرنامج كامل مساحة الشاشة، وبمعنى آخر، تم تكبيرها Maximized. لماذا؟

كما رأينا في الخطوة قبل السابقة، فإن إرسال المفتاحين Alt+Space، يظهران قائمة النظام لبرنامج المقصد، أما الحرف (ك)، فيمثل اختيار البند **تكبير** من قائمة النظام.

▫جرب الخطوة الأخيرة، مع كتابة الحرف (ص)، بدلاً من الحرف (ك)، يمثل الحرف (ص) اختيار البند **تصغير** من قائمة النظام. لاحظ كيف تم تصغير نافذة برنامج المقصد Minimized.

▫تمرن على البرنامجين السابقين ثم اخرج من البرنامج.

كيف يعمل برنامج المصدر

يشغل برنامج المصدر عند بدء تنفيذه، برنامج المقصد آلياً، باستخدام التابع Shell()، ويستخدم العبارة Sendkeys، لإرسال ضربات المفاتيح لبرنامج المقصد.

الإجراء Form_Load():

ينفذ هذا الإجراء آلياً، عند بدء تنفيذ برنامج المصدر. ويستخدم التابع Shell()، لتنفيذ البرنامج الآخر، وهو برنامج المقصد:

```
Private Sub Form_Load()
    Dim ID
    'تنفيذ برنامج المقصد
    ChDir App.Path
    ID = Shell("Dest.exe", vbNormalFocus)
End Sub
```

يمثل الوسيط الأول لهذا التابع، اسم الملف التنفيذي المطلوب تشغيله. أما الوسيط الثاني فهو اختياري، بمعنى أنه يمكننا إهماله (اعتباره غير موجود)، على أي حال، يحدد الوسيط الثاني كيفية تنفيذ البرنامج (نمط تنفيذه)، بمعنى هل سينفذ بشكل طبيعي Normal، أم بشكل مكبر Maximized، أم بشكل مصغر Minimized، أم بشكل مخفي Hidden.

أسندنا قيمة الثابت vbNormalFocus، التي تعني تنفيذ البرنامج بشكل طبيعي Normal، ومع تنشيطه في نفس الوقت.

ملاحظة

إذا لم يحدَد المسار الكامل للملف التنفيذي في الوسيط الأول للتابع ()Shell، فإن التابع ()Shell يفترض وجود الملف التنفيذي في نفس الدليل الحالي، أو في الأدلة المعرفة في بيئة Dos من خلال السطر Path في الملف Autoexec.bat.

الإجراء ()cmdSend_Click

ينفذ هذا الإجراء آلياً عند نقر الزر **إرسال**:

```
Private Sub cmdSend_Click()
    'جعل برنامج المقصد فعالا
    AppActivate "برنامج المقصد"
    'إرسال الأحرف إلى برنامج المقصد
    SendKeys txtUserArea.Text, True
End Sub
```

ينشّط هذا الإجراء برنامج المقصد، من خلال العبارة AppActive:

```
    AppActivate "برنامج المقصد"
```

يمثل الوسيط الأول لهذه العبارة، عنوان النافذة المراد تنشيطها، ويجب مطابقة هذا العنوان تماماً، لما هو مكتوب في شريط عنوان النافذة المراد تنشيطها، وإلا حصل خطأ في مرحلة التنفيذ run time error.

ترسل العبارة التالية Sendkeys، أية ضربة أو ضربات متتالية للمفاتيح، إلى البرنامج النشط الحالي، (هذا يفسر قيامك بتنشيط برنامج المقصد، قبل إرسال ضربات المفاتيح)، لأننا نريد إرسال ضربات المفاتيح لبرنامج المقصد.

بعد انتهاء إرسال ضربات المفاتيح لبرنامج المقصد، يظل برنامج المقصد نشطاً أو فعالاً، وحتى تعيد الفعالية لبرنامج المصدر، أضف للإجراء السابق، العبارة التالية المسؤولة عن تنشيط برنامج ما:

```
    AppActivate "برنامج المصدر"
```

بعد تعليمة Sendkeys مباشرة، ليصبح الإجراء cmdSend_Click() كالتالي:

```
Private Sub cmdSend_Click()
    'جعل برنامج المقصد فعالا
    AppActivate "برنامج المقصد"
    'إرسال الأحرف إلى برنامج المقصد
    SendKeys txtUserArea.Text, True
    AppActivate "برنامج المصدر"
End Sub
```

العبارة Sendkeys

تستخدم العبارة Sendkeys كما رأيت في برنامج المصدر، لإرسال الأحرف، ومحاكاة جميع ضربات لوحة المفاتيح الفعلية، ترسل هذه الضربات إلى البرنامج النشط الحالي، (لا يسمح النظام ويندوز بتنشيط أكثر من برنامج واحد فقط في نفس الوقت).

تستخدم العبارة Sendkeys وسيطين: يمثل الوسيط الأول ضربات المفاتيح التي نريد إرسالها للبرنامج النشط، أما الوسيط الثاني للعبارة Sendkeys فيمثل كيفية إرسال هذه المفاتيح، ويتقبل قيمتين فقط هما True أو False.

تُجبر القيمة True، الإجراء cmdSend_Click() على الانتظار، حتى تُعالج جميع ضربات المفاتيح المرسلة (من قبل البرنامج الذي أُرسلت إليه)، قبل تنفيذ العبارة التالية للعبارة Sendkeys.

أما عند وضع القيمة False، يستمر تنفيذ العبارة التي تلي عبارة Sendkeys مباشرة، دون انتظارٍ لمعالجة ضربات المفاتيح المرسلة.

في مثالنا السابق، استخدمنا القيمة True في عبارة Sendkeys. لذلك سوف ينتظر برنامج المصدر، لحين انتهاء برنامج المقصد، من معالجة ضربات المفاتيح التي أرسلت إليه.

أما إذا وضعت القيمة False بدلاً من القيمة True، فسوف يعود التحكم إلى برنامج المصدر مباشرة، بعد إرسال ضربات المفاتيح، دون انتظار برنامج المقصد للانتهاء من عمله.

يمكن استخدام العبارة Sendkeys لإرسال مفاتيح خاصة مذكورة في الجدول 18-3. مثل إرسال المفتاح Alt+ع، بكتابة "%ع".

عند ضغط المفتاحين Alt+ع في برنامج المقصد، ستحصل على نفس النتيجة، كما لو نقرت الزر **عرض الرسالة**. بسبب أن قيمة الخاصية Caption للزر **عرض الرسالة** هي "&عرض الرسالة"، والتي تعني أن الحرف (ع) يمثل حرف الوصول السريع Hotkey للزر **عرض الرسالة**.

جدول 18-3. شفرة الأحرف الخاصة المستخدمة مع عبارة Sendkeys.

المفتاح	طريقة كتابته في العبارة Sendkeys
SHIFT	+
CTRL	^
ALT	%
BACKSPACE	{BACKSPACE}, {BS}, or {BKSP}
BREAK	{BREAK}
CAPS LOCK	{CAPSLOCK}
DEL or DELETE	{DELETE} or {DEL}
DOWN ARROW	{DOWN}
END	{END}
ENTER	{ENTER}or ~
ESC	{ESC}

HELP	{HELP}
HOME	{HOME}
INS or INSERT	{INSERT} or {INS}
LEFT ARROW	{LEFT}
NUM LOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
PRINT SCREEN	{PRTSC}
RIGHT ARROW	{RIGHT}
SCROLL LOCK	{SCROLLLOCK}
TAB	{TAB}
UP ARROW	{UP}
F1	{F1}
F2	{F2}
...	...
F16	{F16}

كيف يُرسِل البرنامج ضربات المفاتيح إلى نفسه؟

برنامج الرنين

ترسل العبارة Sendkeys ضربات المفاتيح للبرنامج النشط الحالي، ماذا يحصل لو كان برنامج المصدر هو البرنامج النشط، وقت إرسال ضربات المفاتيح؟. كما توقعت تماماً فهذه الضربات، سترسل إلى برنامج المصدر نفسه!

هل يوجد استخدامات خاصة لعملية إرسال الضربات إلى البرنامج نفسه؟. بالتأكيد يوجد، مثال على ذلك، برامج العرض أو البرامج التدريبية، حيث تقوم هذه البرامج بتشغيل نفسها، بغرض شرح كيفية عملها للمستخدم، أو تدريبه على العمل على هذه البرامج. وسوف يشرح برنامج الرنين كيفية عمل ذلك في فيجول بيسك.

التصميم المرئي لبرنامج الرنين:

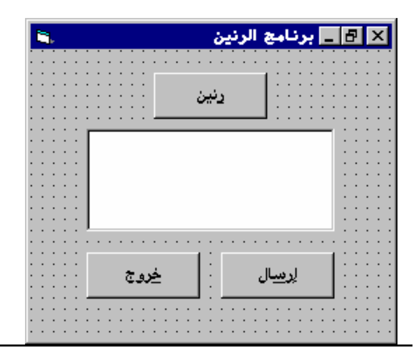
□ابدأ مشروعاً جديداً من النوع Standard EXE واحفظ المشروع في الدليل C:\VB5Prg\Ch18 باسم MySelf.Vbp، واحفظ النموذج الجديد باسم MySelf.Frm في نفس الدليل.

▫أنشئ النموذج frmMySelf وفقاً للجدول 18-4.

بعد انتهاء النموذج، ينبغي أن يصبح كما في الشكل 18-6.

جدول 18-4. جدول خصائص النموذج frmMySelf.

الكائن	الخاصية	القيمة
Form	**Name**	**frmDest**
	Caption	برنامج الرنين
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdBeep**
	Caption	&رنين
	RightToLeft	True
CommandButton	**Name**	**cmdSend**
	Caption	إر&سال
	RightToLeft	True
TextBox	**Name**	**txtUserArea**
	MultiLine	True
	Text	(اجعله فارغاً)
	RightToLeft	True



الشكل 18-6

نموذج برنامج الرنين

في مرحلة التصميم.

كتابة نص برنامج الرنين

- أدخل النص التالي في قسم التصريحات العامة للنموذج frmMySelf:

```
'يجب التصريح عن كل المتحولات في البرنامج
Option Explicit
```

- أدخل النص التالي في الإجراء cmdSend_Click():

```
Private Sub cmdSend_Click()
    'تأكيد نشاط برنامج الرنين
    AppActivate "برنامج الرنين"
    'إرسال النص المكتوب في مربع النص
    'للبرنامج نفسه
    SendKeys txtUserArea, True
End Sub
```

- أدخل النص التالي في الإجراء cmdBeep_Click():

```
Private Sub cmdBeep_Click()
    Beep
    MsgBox "لقد نقرت زر رنين"
End Sub
```

- أدخل النص التالي في الإجراء cmdExit_Click():

```
Private Sub cmdExit_Click()
    End
End Sub
```

تنفيذ برنامج الرنين

نفّذ برنامج الرنين كالتالي:

- اختر البند **...Make MySelf.exe**، من القائمة **File**، واحفظ الملف التنفيذي MySelf.exe، في الدليل C:\VB5Prg\EXE.
- أنهِ عمل فيجول بيسك، من القائمة **File** البند **Exit**.
- نفّذ البرنامج MySelf.exe.
- انقر الزر **رنين**.

ينبغي على جهازك إصدار صوت، ولمدة قصيرة (لا ضرورة لوجود بطاقة صوت

داخل جهازك، لأن الصوت يصدر عن مكبر الصوت الداخلي).

- اضغط المفتاحين ر+Alt في لوحة المفاتيح.

*ينبغي على جهازك أيضاً، إصدار صوت، لأن عنوان الزر **رنين** هو "&رنين"، والحرف (ر) يمثل حرف الوصول السريع له.*

- اكتب %ر في مربع النص التابع لبرنامج الرنين، ثم انقر الزر **إرسال**.

ينبغي على جهازك أيضاً، إصدار صوت رنين.

- تدرب على برنامج الرنين ثم اخرج منه.

نص الإجراء ()cmdSend_Click

ينفذ هذا الإجراء آلياً، عند نقر الزر **إرسال** في برنامج الرنين:

تقوم العبارة الأولى في هذا الإجراء، على تأكيد نشاط برنامج الرنين. في الحقيقة، لا داعي لهذه العبارة برمجياً، لأنك بمجرد نقر الزر **رنين**، يُنشط البرنامج آلياً.

أما العبارة التالية Sendkeys، فهي ترسل النص المكتوب في مربع النص، إلى البرنامج النشط الحالي (برنامج الرنين)، وكأنه مرسل من لوحة المفاتيح الفعلية.

إذا كتبت %ر في مربع النص، فإنه يُترجم على أنه المفتاح ر+Alt، (كما هو واضح في الشكل 18−6).

وهو يشبه عملية نقر الزر **رنين**.

يحتوي الإجراء ()cmdBeep_Click العبارتين Beep و MsgBox، اللتين تتسببان في إصدار صوت من المكبر الداخلي لجهازك، ثم إظهار مربع رسالة.

الخلاصة

تعلمت فى هذا الفصل كيفية إستخدام الوظسفة () Shell، لتنفيذ برنامج أخر من خلال برنامجك، وكيفية إرسال المعلومات من تطبيق لأخر، أو من التطبيق وإليه، عبر العبارة Sendkeys.

الفصل الثامن عشر

محاكاة ضربات المفاتيح

سوف تتعلم في هذا الفصل، كيفية إرسال ضربات المفاتيح عبر برنامجك إلى برامج ويندوز الأخرى، وكيفية توجيه ضربات المفاتيح من البرنامج الذي أنشأته إلى البرنامج نفسه مرة أخرى، تفيد هذه التقنية في إنشاء برامج دعائية Demo أو أية برامج أخرى.

محاكاة ضربات المفاتيح: برنامجي المصدر والمقصد

تستطيع كتابة برنامج ما، عبر لغة فيجول بيسك، يُرسل ضربات المفاتيح، وكأنها أرسلت من لوحة المفاتيح Keyboard الفعلية. سنكتب الآن برنامجين، Source.exe و Dest.exe. يولد البرنامج Source.exe ضربات المفاتيح، ويستقبل البرنامج Dest.exe هذه الضربات المنشأة من البرنامج Source.exe.

التصميم المرئي لبرنامج المصدر

- ابدأ مشروعاً جديداً من النوع Standard EXE، واحفظ المشروع باسم Source.Vbp في الدليل C:\VB5Prg\Ch18، واحفظ النموذج باسم Source.frm في نفس الدليل.
- أنشىء النموذج frmSource وفقاً للجدول 18-1.

جدول 18-1. جدول خصائص النموذج frmSource.

الكائن	الخاصية	القيمة
Form	**Name**	**frmSource**
	Caption	برنامج المصدر
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdSend**
	Caption	إر&سال

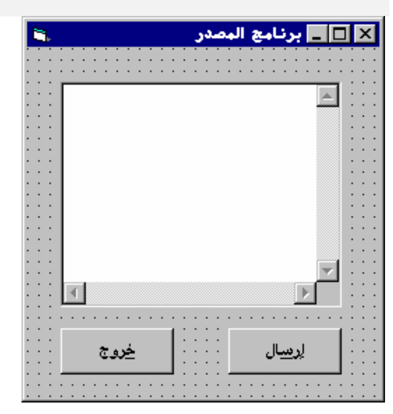
	RightToLeft	True
TextBox	**Name**	**txtUserArea**
	MultiLine	True
	ScrollBars	3-Both
	RightToLeft	True

بعد انتهاء النموذج ينبغي أن يصبح كما في الشكل 18-1.

إدخال نص البرنامج frmSource

- أدخل النص التالي في قسم التصريحات العامة للنموذج frmSource:

```
'يجب التصريح عن كل المتحولات في البرنامج
Option Explicit
```



الشكل 18-1

النموذج frmSource بعد انتهاء تصميمه.

- أدخل النص التالي في الإجراء ()Form_Load:

```
Private Sub Form_Load()
    Dim ID
    'تنفيذ برنامج المقصد
    ChDir App.Path
    ID = Shell("Dest.exe", vbNormalFocus)
End Sub
```

- أدخل النص التالي في الإجراء ()cmdSend_Click:

```
Private Sub cmdSend_Click()
    'جعل برنامج المقصد فعالا
    AppActivate "برنامج المقصد"
```

```
    'إرسال الأحرف إلى برنامج المقصد
    SendKeys txtUserArea.Text, True
End Sub
```

- أدخل النص التالي في الإجراء ()cmdExit_Click:

```
Private Sub cmdExit_Click()
    End
End Sub
```

- احفظ المشروع.
- أنشئ الملف التنفيذي Source.exe، باختيار البند **...Make Source.exe** من القائمة **File**، واحفظه في الدليل C:\VB5Prg\EXE.

لا تستطيع تنفيذ البرنامج Source.exe بعدُ، لأنه يحتاج إلى البرنامج Dest.exe لكي يعمل بصورة طبيعية.

التصميم المرئي لبرنامج المقصد

- ابدأ مشروعاً جديداً من النوع Standard EXE، واحفظ المشروع في الدليل C:\VB5Prg\Ch18 باسم Dest.Vbp، واحفظ النموذج الجديد باسم Dest.Frm في نفس الدليل.
- أنشأ النموذج frmDest وفقاً للجدول 18−2.

جدول 18−2. جدول خصائص النموذج frmDest.

الكائن	الخاصية	القيمة
Form	**Name**	**frmDest**
	Caption	برنامج المقصد
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdDisplayMessage**

&عرض الرسالة	Caption	
True	RightToLeft	

بعد انتهاء النموذج ينبغي أن يصبح كما في الشكل 18-2.



الشكل 18-2

النموذج frmDest بعد انتهاء تصميمه.

إدخال نص برنامج المقصد

- أدخل النص التالي في قسم التصريحات العامة للنموذج frmDest:

```
' يجب التصريح عن كل المتحولات في البرنامج
Option Explicit
```

- أدخل النص التالي في الإجراء ()cmdDisplayMessage_Click:

```
Private Sub cmdDisplayMessage_Click()
    MsgBox "لقد نقرت زر عرض الرسالة"
End Sub
```

- أدخل النص التالي في الإجراء ()cmdExit_Click:

```
Private Sub cmdExit_Click()
    End
End Sub
```

سوف ننشئ الآن الملف التنفيذي Dest.exe:

- اختر البند **...Make Dest.exe** من القائمة **File**، واحفظ الملف التنفيذي Dest.exe في الدليل C:\VB5Prg\EXE.
- تأكد من وجود الملفين التنفيذيين Source.exe و Dest.exe في نفس الدليل السابق.

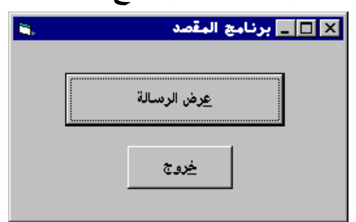
تنفيذ برنامجي المصدر والمقصد معاً

نفّذ أولاً البرنامج Dest.exe:

- أنهِ فيجول بيسك، من القائمة **File**، البند **Exit**

- افتح الدليل C:\VB5Prg\EXE من المستكشف، ثم نفّذ البرنامج Dest.exe، بالنقر المزدوج على رمزه، كأي برنامج آخر.

تظهر نافذة البرنامج Dest.exe كما في الشكل 18-3.



الشكل 18-3

نافذة برنامج المقصد.

- انقر الزر **عرض الرسالة** في البرنامج Dest.exe.

تظهر الرسالة التالية "لقد نقرت زر عرض الرسالة " كما في الشكل 18-4.



الشكل 18-4

الرسالة التي تظهر عقب نقر الزر **عرض الرسالة**.

- انقر الزر **خروج** في البرنامج Dest.exe لإنهائه.

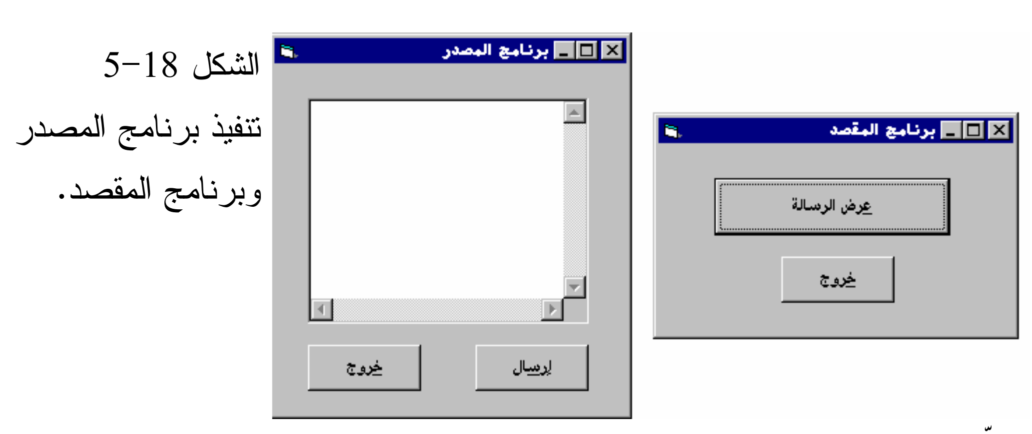
لاحظ بساطة برنامج المقصد، واحتوائه على زرين فقط، أحدهما لإظهار رسالة، والثاني للخروج.

الهدف من البرنامجين السابقين، توضيح فكرة استطاعتك نقر الزر **عرض الرسالة** الموجود في برنامج المقصد، من خلال برنامج المصدر.

اتبع الخطوات التالية لرؤية ذلك فعلياً:

- تأكد أن البرنامج Dest.exe غير منفذ حالياً.
- نفّذ البرنامج Source.exe.

يتم تنفيذ البرنامج Source.exe، وآلياً سوف يتم تنفيذ البرنامج Dest.exe كما في الشكل 18-5.



الشكل 18-5

تنفيذ برنامج المصدر وبرنامج المقصد.

صغّر كافة النوافذ ماعدا نافذتي البرنامجين السابقين، وضعهما بشكل متجاور، بحيث تراهما بشكل واضح ومنفصل.

- اكتب %ع (التي تعني ضغط المفتاحين Alt+ع في لوحة المفاتيح) في مربع النص الموجود في برنامج المصدر.
- انقر الزر **إرسال** في برنامج المصدر، لإرسال المفتاحين السابقين إلى برنامج المقصد.

كما ترى، أظهر برنامج المقصد الرسالة الخاصة به، وكأن أحدهم قد نقر الزر **عرض الرسالة**، (لم يفعل أحد ذلك)، وإنما قلّد البرنامج Source.exe إرسال ضربات المفاتيح، وكأنها أرسلت من لوحة المفاتيح فعلياً.

- اكتب هذه المرة %خ (التي تعني ضغط المفتاحين Alt+خ في لوحة المفاتيح) في مربع النص الموجود في برنامج المصدر (تذكر مسح محتويات مربع النص أولاً).
- انقر الزر **إرسال** في برنامج المصدر، لإرسال المفتاحين السابقين إلى برنامج المقصد.

ماذا حصل، لقد تم إنهاء برنامج المقصد، عن طريق نقر الزر **خروج**، وأيضاً لم يفعل أحد ذلك، وإنما تم هذا الأمر عن طريق برنامج المصدر، الذي أرسل ضربتي المفتاحين Alt+خ إلى برنامج المقصد، وبدوره أُجبر على الخروج.

- جرب كتابة % (إضافة لحرف مسافة واحد بعده)، التي تعني ضغط المفتاحين Alt+Space في لوحة المفاتيح.

- انقر الزر **إرسال** في برنامج المصدر، لإرسال المفتاحين السابقين إلى برنامج المقصد.

لقد ظهرت قائمة النظام System Menu، في برنامج المقصد.

- أضف للكتابة السابقة (%حرف مسافة واحد)، الحرف (ك)، لتصبح (% ك)، ثم أعد نقر الزر **إرسال**.

لقد أخذت نافذة البرنامج كامل مساحة الشاشة، وبمعنى آخر، تم تكبيرها Maximized. لماذا؟

كما رأينا في الخطوة قبل السابقة، فإن إرسال المفتاحين Alt+Space، يظهران قائمة النظام لبرنامج المقصد، أما الحرف (ك)، فيمثل اختيار البند **تكبير** من قائمة النظام.

- جرب الخطوة الأخيرة، مع كتابة الحرف (ص)، بدلاً من الحرف (ك)، يمثل الحرف (ص) اختيار البند **تصغير** من قائمة النظام. لاحظ كيف تم تصغير نافذة برنامج المقصد Minimized.
- تمرن على البرنامجين السابقين ثم اخرج من البرنامج.

كيف يعمل برنامج المصدر

يشغل برنامج المصدر عند بدء تنفيذه، برنامج المقصد آلياً، باستخدام التابع Shell()، ويستخدم العبارة Sendkeys، لإرسال ضربات المفاتيح لبرنامج المقصد.

الإجراء Form_Load():

ينفذ هذا الإجراء آلياً، عند بدء تنفيذ برنامج المصدر. ويستخدم التابع Shell()، لتنفيذ البرنامج الآخر، وهو برنامج المقصد:

```
Private Sub Form_Load()
    Dim ID
    'تنفيذ برنامج المقصد
    ChDir App.Path
    ID = Shell("Dest.exe", vbNormalFocus)
End Sub
```

يمثل الوسيط الأول لهذا التابع، اسم الملف التنفيذي المطلوب تشغيله. أما الوسيط الثاني فهو اختياري، بمعنى أنه يمكننا إهماله (اعتباره غير موجود)، على أي حال، يحدد الوسيط الثاني كيفية تنفيذ البرنامج (نمط تنفيذه)، بمعنى هل سينفذ بشكل طبيعي Normal، أم بشكل مكبر Maximized، أم بشكل مصغر Minimized، أم بشكل مخفي Hidden.

أسندنا قيمة الثابت vbNormalFocus، التي تعني تنفيذ البرنامج بشكل طبيعي Normal، ومع تنشيطه في نفس الوقت.

ملاحظة

إذا لم يحدَد المسار الكامل للملف التنفيذي في الوسيط الأول للتابع ()Shell، فإن التابع ()Shell يفترض وجود الملف التنفيذي في نفس الدليل الحالي، أو في الأدلة المعرفة في بيئة Dos من خلال السطر Path في الملف Autoexec.bat.

الإجراء ()cmdSend_Click

ينفذ هذا الإجراء آلياً عند نقر الزر **إرسال**:

```
Private Sub cmdSend_Click()
    'جعل برنامج المقصد فعالا
    AppActivate "برنامج المقصد"
    'إرسال الأحرف إلى برنامج المقصد
    SendKeys txtUserArea.Text, True
End Sub
```

ينشّط هذا الإجراء برنامج المقصد، من خلال العبارة AppActive:

```
    AppActivate "برنامج المقصد"
```

يمثل الوسيط الأول لهذه العبارة، عنوان النافذة المراد تنشيطها، ويجب مطابقة هذا العنوان تماماً، لما هو مكتوب في شريط عنوان النافذة المراد تنشيطها، وإلا حصل خطأ في مرحلة التنفيذ run time error.

ترسل العبارة التالية Sendkeys، أية ضربة أو ضربات متتالية للمفاتيح، إلى البرنامج النشط الحالي، (هذا يفسر قيامك بتنشيط برنامج المقصد، قبل إرسال ضربات المفاتيح)، لأننا نريد إرسال ضربات المفاتيح لبرنامج المقصد.

بعد انتهاء إرسال ضربات المفاتيح لبرنامج المقصد، يظل برنامج المقصد نشطاً أو فعالاً، وحتى تعيد الفعالية لبرنامج المصدر، أضف للإجراء السابق، العبارة التالية المسئولة عن تنشيط برنامج ما:

```
AppActivate "برنامج المصدر"
```

بعد تعليمة Sendkeys مباشرة، ليصبح الإجراء cmdSend_Click() كالتالي:

```
Private Sub cmdSend_Click()
    'جعل برنامج المقصد فعالا
    AppActivate "برنامج المقصد"
    'إرسال الأحرف إلى برنامج المقصد
    SendKeys txtUserArea.Text, True
    AppActivate "برنامج المصدر"
End Sub
```

العبارة Sendkeys

تستخدم العبارة Sendkeys كما رأيت في برنامج المصدر، لإرسال الأحرف، ومحاكاة جميع ضربات لوحة المفاتيح الفعلية، ترسل هذه الضربات إلى البرنامج النشط الحالي، (لا يسمح النظام ويندوز بتنشيط أكثر من برنامج واحد فقط في نفس الوقت).

تستخدم العبارة Sendkeys وسيطين: يمثل الوسيط الأول ضربات المفاتيح التي نريد إرسالها للبرنامج النشط، أما الوسيط الثاني للعبارة Sendkeys فيمثل كيفية إرسال هذه المفاتيح، ويتقبل قيمتين فقط هما True أو False.

تُجبر القيمة True، الإجراء cmdSend_Click() على الانتظار، حتى تُعالج جميع ضربات المفاتيح المرسلة (من قبل البرنامج الذي أُرسلت إليه)، قبل تنفيذ العبارة التالية للعبارة Sendkeys.

أما عند وضع القيمة False، يستمر تنفيذ العبارة التي تلي عبارة Sendkeys مباشرة، دون انتظارٍ لمعالجة ضربات المفاتيح المرسلة.

في مثالنا السابق، استخدمنا القيمة True في عبارة Sendkeys. لذلك سوف ينتظر برنامج المصدر، لحين انتهاء برنامج المقصد، من معالجة ضربات المفاتيح التي أرسلت إليه.

أما إذا وضعت القيمة False بدلاً من القيمة True، فسوف يعود التحكم إلى برنامج المصدر مباشرة، بعد إرسال ضربات المفاتيح، دون انتظار برنامج المقصد للانتهاء من عمله.

يمكن استخدام العبارة Sendkeys لإرسال مفاتيح خاصة مذكورة في الجدول 18-3. مثل إرسال المفتاح Alt+ع، بكتابة "%ع".

عند ضغط المفتاحين Alt+ع في برنامج المقصد، ستحصل على نفس النتيجة، كما لو نقرت الزر **عرض الرسالة**. بسبب أن قيمة الخاصية Caption للزر **عرض الرسالة** هي "&عرض الرسالة"، والتي تعني أن الحرف (ع) يمثل حرف الوصول السريع Hotkey للزر **عرض الرسالة**.

جدول 18-3. شفرة الأحرف الخاصة المستخدمة مع عبارة Sendkeys.

طريقة كتابته في العبارة Sendkeys	المفتاح
+	SHIFT
^	CTRL
%	ALT
{BACKSPACE}, {BS}, or {BKSP}	BACKSPACE
{BREAK}	BREAK
{CAPSLOCK}	CAPS LOCK
{DELETE} or {DEL}	DEL or DELETE
{DOWN}	DOWN ARROW
{END}	END
{ENTER}or ~	ENTER
{ESC}	ESC
{HELP}	HELP

HOME	{HOME}
INS or INSERT	{INSERT} or {INS}
LEFT ARROW	{LEFT}
NUM LOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
PRINT SCREEN	{PRTSC}
RIGHT ARROW	{RIGHT}
SCROLL LOCK	{SCROLLLOCK}
TAB	{TAB}
UP ARROW	{UP}
F1	{F1}
F2	{F2}
...	...
F16	{F16}

كيف يُرسِل البرنامج ضربات المفاتيح إلى نفسه؟

برنامج الرنين

ترسل العبارة Sendkeys ضربات المفاتيح للبرنامج النشط الحالي، ماذا يحصل لو كان برنامج المصدر هو البرنامج النشط، وقت إرسال ضربات المفاتيح؟. كما توقعت تماماً فهذه الضربات، سترسل إلى برنامج المصدر نفسه!

هل يوجد استخدامات خاصة لعملية إرسال الضربات إلى البرنامج نفسه؟. بالتأكيد يوجد، مثال على ذلك، برامج العرض أو البرامج التدريبية، حيث تقوم هذه البرامج بتشغيل نفسها، بغرض شرح كيفية عملها للمستخدم، أو تدريبه على العمل على هذه البرامج. وسوف يشرح برنامج الرنين كيفية عمل ذلك في فيجول بيسك.

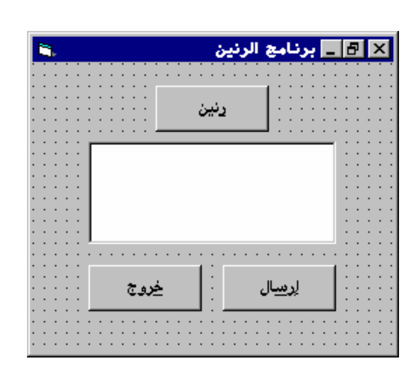
التصميم المرئي لبرنامج الرنين:

- ابدأ مشروعاً جديداً من النوع Standard EXE واحفظ المشروع في الدليل C:\VB5Prg\Ch18 باسم MySelf.Vbp، واحفظ النموذج الجديد باسم MySelf.Frm في نفس الدليل.
- أنشئ النموذج frmMySelf وفقاً للجدول 18-4.

بعد انتهاء النموذج، ينبغي أن يصبح كما في الشكل 18-6.

جدول 18-4. جدول خصائص النموذج frmMySelf.

الكائن	الخاصية	القيمة
Form	**Name**	**frmDest**
	Caption	برنامج الرنين
	RightToLeft	True
CommandButton	**Name**	**cmdExit**
	Caption	&خروج
	RightToLeft	True
CommandButton	**Name**	**cmdBeep**
	Caption	&رنين
	RightToLeft	True
CommandButton	**Name**	**cmdSend**
	Caption	إر&سال
	RightToLeft	True
TextBox	**Name**	**txtUserArea**
	MultiLine	True
	Text	(اجعله فارغاً)
	RightToLeft	True



الشكل 18-6
نموذج برنامج الرنين
في مرحلة التصميم.

كتابة نص برنامج الرنين

- أدخل النص التالي في قسم التصريحات العامة للنموذج frmMySelf:

```
'يجب التصريح عن كل المتحولات في البرنامج
Option Explicit
```

- أدخل النص التالي في الإجراء ()cmdSend_Click:

```
Private Sub cmdSend_Click()
    'تأكيد نشاط برنامج الرنين
    AppActivate "برنامج الرنين"
    'إرسال النص المكتوب في مربع النص
    'للبرنامج نفسه
    SendKeys txtUserArea, True
End Sub
```

- أدخل النص التالي في الإجراء ()cmdBeep_Click:

```
Private Sub cmdBeep_Click()
    Beep
    MsgBox "لقد نقرت زر رنين"
End Sub
```

- أدخل النص التالي في الإجراء ()cmdExit_Click:

```
Private Sub cmdExit_Click()
    End
End Sub
```

تنفيذ برنامج الرنين

نفّذ برنامج الرنين كالتالي:

- اختر البند **...Make MySelf.exe**، من القائمة **File**، واحفظ الملف التنفيذي MySelf.exe، في الدليل C:\VB5Prg\EXE.
- أنهِ عمل فيجول بيسك، من القائمة **File** البند **Exit**.
- نفّذ البرنامج MySelf.exe.
- انقر الزر **رنين**.

ينبغي على جهازك إصدار صوت، ولمدة قصيرة (لا ضرورة لوجود بطاقة صوت داخل جهازك، لأن الصوت يصدر عن مكبر الصوت الداخلي).

- اضغط المفتاحين ر+Alt في لوحة المفاتيح.

*ينبغي على جهازك أيضاً، إصدار صوت، لأن عنوان الزر **رنين** هو "&رنين"، والحرف (ر) يمثل حرف الوصول السريع له.*

- اكتب %ر في مربع النص التابع لبرنامج الرنين، ثم انقر الزر **إرسال**.

ينبغي على جهازك أيضاً، إصدار صوت رنين.

- تدرب على برنامج الرنين ثم اخرج منه.

نص الإجراء ()cmdSend_Click

ينفذ هذا الإجراء آلياً، عند نقر الزر **إرسال** في برنامج الرنين:

تقوم العبارة الأولى في هذا الإجراء، على تأكيد نشاط برنامج الرنين. في الحقيقة، لا داعي لهذه العبارة برمجياً، لأنك بمجرد نقر الزر **رنين**، يُنشط البرنامج آلياً.

أما العبارة التالية Sendkeys، فهي ترسل النص المكتوب في مربع النص، إلى البرنامج النشط الحالي (برنامج الرنين)، وكأنه مرسل من لوحة المفاتيح الفعلية.

إذا كتبت %ر في مربع النص، فإنه يُترجم على أنه المفتاح ر+Alt، (كما هو واضح في الشكل 18−6).

وهو يشبه عملية نقر الزر **رنين**.

يحتوي الإجراء ()cmdBeep_Click على العبارتين Beep و MsgBox، اللتين تتسببان في إصدار صوت من المكبر الداخلي لجهازك، ثم إظهار مربع رسالة.

الخلاصة

تعلمت فى هذا الفصل كيفية إستخدام الوظسفة () Shell، لتنفيذ برنامج أخر من خلال برنامجك، وكيفية إرسال المعلومات من تطبيق لأخر، أو من التطبيق وإليه، عبر العبارة Sendkeys.

الفصل التاسع عشر

تقنية ActiveX، الأصوات، الوسائط المتعددة

تعلمت من الفصول السابقة، أساسيات لغة فيجول بيسك. مثل إنشاء النماذج، ووضع الكائنات على النموذج، وربط نص برنامج معين بهذه الكائنات. يركز هذا الفصل على كيفية تصميم التطبيقات في ويندوز مع تقنية ActiveX.

تقنية ActiveX

تعتبر تقنية ActiveX تقنية مهمة جداً، ويعتمد عليها ويندوز بشكل كبير، وهي بتعريف بسيط: إمكانية تشغيل وحدة نمطية تنفيذية EXE Module، التي تمثل برنامجاً كاملاً، أو جزء من برنامج، أو عنصر تحكم، داخل برنامجك الرئيسي.

بكلام آخر، تضمين برنامجٍ آخر في برنامجك، بدون كتابة سطر إضافي من نص البرنامج.

مثلاً، لو أردت تصميم برنامجٍ، يُشغل ملفات الوسائط المتعددة، مثل ملفات الصوت Wave، أو ملفات الأفلام والحركة AVI، أو ملفات الموسيقى MIDI، لاحتجت إلى وقت طويل جداً لكتابة هذا البرنامج، مع استخدام الكثير من توابع API، (هذا إذا استطعت كتابته أصلاً)، هذا بالإضافة إلى إهدار الكثير من الجهد والوقت.

بدلاً من ذلك، يمكنك الاستفادة من برنامج جاهز (قابل للدمج مع برنامجك الرئيسي)، يقوم بنفس العمل الذي تود تنفيذه. تصور الكسب في الوقت والجهد الذي اختصرته لبناء تطبيقك الرئيسي، وصرف هذا الوقت والجهد عليه.

يشبه هذا الأمر عملية بناء منزل مثلاً، وطلبت من المتعهد **فيجول بيسك**، بناء هذا المنزل تحت إشرافك الشخصي، هنا لديك احتمالين:

- إما أن تبني المنزل كاملاً بنفسك، بمساعدة المتعهد **فيجول بيسك**.
- أو شراء بعض الأشياء الجاهزة من شركات أخرى، كالأبواب والنوافذ والأثاث مثلاً.

تصور الوقت المختصر في الاحتمال الثاني، الذي ينعكس على سرعة الإنجاز. ولا بد أنك تخيلت الآن طريقة عمل تقنية ActiveX، وما توفره من ميزات كثيرة.

في الحقيقة، لقد استعملت هذه التقنية منذ بداية هذا الكتاب.

فزر الأمر Command Button مثلاً، هو عنصر تحكم ActiveX، وجميع الأدوات التي استخدمتها سابقاً هي عناصر تحكم ActiveX.تسمى هذه العناصر بالعناصر القياسية التي تأتي أصلاً مع لغة فيجول بيسك، وهي ليست ملفات منفصلة بل موجودة في نواة اللغة نفسها.

تأتي بعض العناصر الأخرى مع فيجول بيسك أيضاً، ولكنها منفصلة عنه (موجودة في ملف منفصل بامتداد OCX).

لو عدنا لزر الأمر، لوجدنا أن له عملاً معيناً، هو تنفيذ نص برنامج ما، عند النقر عليه، وهو ما فعلناه في معظم أمثلة هذا الكتاب، لكن ألم تسأل نفسك، كيف يغير هذا الزر شكله عند النقر عليه، وأين هي أسطر البرنامج المسئولة عن فعل ذلك؟. قس هذا الأمر على باقي الأدوات والعناصر الأخرى.

للإجابة على هذا السؤال نقول: تعتبر هذه العناصر في الحقيقة، عناصر تحكم تعمل بتقنية ActiveX، وهي عبارة عن وحدة نمطية تنفيذية، لا تعمل بشكل منفصل، إنما ضمن تطبيق رئيسي. (مثل الباب في مثالنا السابق عن بناء المنزل، فالباب بحد ذاته، لا يركب بدون منزل).

دعنا الآن نلخص مزايا استخدام تقنية ActiveX:

- ***تطوير أسرع للبرامج:*** يوفر استخدام عناصر تحكم ActiveX الوقت اللازم لبناء البرنامج الرئيسي، بدلاً من ضياعه في توفير مزايا موجودة أصلاً. بكلام آخر، أنت لست مضطراً لاختراع العجلة من جديد.
- ***ثقة أعلى لتطبيقاتك:*** طورت عناصر التحكم ActiveX التي تستخدمها، شركات خاصة، مهمتها توفير هذه العناصر وبيعها، تحت إشراف مبرمجين مختصين. هذا الأمر يضمن لك وثوقية عالية في تطبيقاتك، التي تعني عدم حصول أخطاء غير متوقعة، لأن هذه العناصر قد اختبرت بشكل كامل، وهي

خالية تقريباً من الأخطاء والشوائب.

▪ ***تقليل الوقت اللازم للتعلم:*** تعمل جميع عناصر تحكم ActiveX بنفس الطريقة تقريباً، وبمجرد تعلم مبدأ عمل تقنية ActiveX، تستطيع استخدام باقي العناصر، مع القليل من معرفة خصوصية كل عنصر بالذات. بالإضافة إلى أنك قد تستخدم عنصر تحكم ما، وأنت لا تعرف بالضبط، كيفية عمل هذا العنصر داخلياً (في الحقيقة، لا يهمك كثيراً كيفية عمله). وبذلك تكون قد وفرت وقتاً طويلاً في تعلم كيف يقوم هذا العنصر بعمله، لو أردت تنفيذ هذا العمل بنفسك.

▪ ***واجهة استخدام مألوفة للمستخدم:*** يجعل استخدام عناصر التحكم ActiveX القياسية، برنامجك أو تطبيقك ذو واجهة استخدام مألوفة من قبل المستخدم، وهي مشابهة لباقي واجهات التطبيقات الأخرى، التي اعتاد وتدرب المستخدم عليها. تصور لو أنك أنشأت زر أمر خاص بك، ومختلف عن زر الأمر القياسي، توقع عندئذ أن المستخدم قد لا يعرف أن هذا الشيء الذي أنشأته هو زر أمر.

قواعد استخدام عناصر ActiveX

هناك بعض القواعد الرئيسية التي يجب اتباعها عند استخدام أدوات ActiveX:

- لا تسرف في استخدام أدوات ActiveX، بل استخدم ما يلزمك منها فقط. لأن كثرة الأدوات في تطبيقاتك، سوف تبطئ عمل التطبيق بشكل ملحوظ، وخصوصاً عند بداية التنفيذ.
- حاول قدر الإمكان، استخدام الأدوات القياسية التي تأتي أصلاً مع لغة فيجول بيسك. وبذلك تحصل على عدة مزايا منها:

▪ ضمان استمرار تطوير هذه العناصر من قبل Microsoft في الإصدارات الجديدة من فيجول بيسك (وهو ما لن تستطيع ضمانه مع الشركات الأخرى، فيما لو استخدمت عناصرها).

▪ تقليل حجم البرنامج الكلي (أقراص الإعداد).

- التأكد أن هذه الأدوات القياسية، تعمل بشكل خال من الأخطاء تقريباً، وبتوافق عالي مع النظام ككل.
- يظهر برنامجك بالنسبة للمستخدم بشكل مألوف (سرعة في التعلم).

- هناك قاعدة تقول: إذا استطعت الاستغناء عن عنصر تحكم ActiveX بقليل من الجهد، فافعل ذلك فوراً. لأن هذا الأمر يزيد من سرعة تنفيذ وثقة تطبيقك بآن واحد.
- لا تستخدم عناصر التحكم ActiveX ذات الإصدار الدعائي أو الاستعراضي Demo Version، بل استخدم العناصر الأصلية الكاملة والمسجلة Register. يسبب استخدام العناصر الدعائية الكثير من المشاكل، وهي نسخة غير قابلة للتوزيع غالباً. طبعاً، الاستخدام الذي قصدته هو الاستخدام النهائي، وليس بقصد تجربة العنصر. هذا يعني أنه يمكنك استخدام العنصر الدعائي لتجربته فقط، ولكن لا تضعه أبداً في أقراص الإعداد.

برنامج الوسائط المتعددة

يوضح لنا برنامج الوسائط المتعددة، كيفية استخدام عنصر التحكم Microsoft Multimedia (اسم الملف هو MCI32.OCX)، لإنشاء برنامج يمكنه تشغيل ملفات الوسائط المتعددة.

يتمكن برنامج الوسائط المتعددة (بالاعتماد على العنصر MCI32.OCX)، من فتح ملفات الصوت ذات الامتداد WAV، وفتح ملفات الحركة ذات الامتداد AVI، وأيضاً فتح ملفات الموسيقى ذات الامتداد MID.

يوفر برنامج الوسائط المتعددة، واجهة استخدام سهلة، وقائمة تمكننا من فتح أحد أنواع الملفات الثلاثة السابقة، بعد فتح الملف يمكننا:

- عزف الملف Play.
- إيقاف عزف الملف Stop.
- إيقاف مؤقت، لعزف الملف Pause.
- إعادة المؤشر في أول الملف Back.
- وضع المؤشر في نهاية الملف End.

- تقديم الملف صورة للأمام Next Frame.
- ترجيع الملف صورة للخلف Previous Frame.انظر الشكل 19-1.



الشكل 19-1

برنامج الوسائط المتعددة.

كما يوفر برنامج الوسائط المتعددة، بعض المعلومات عن الملف الحالي تتلخص فيما يلي:

- اسم ومسار الملف الحالي المفتوح.
- طول الملف الحالي (تتغير وحدة الطول حسب نوع الملف).
- موقع العزف الحالي من الملف (إظهاره كرقم صحيح).
- مؤشر يدلنا على موقع العزف الحالي في الملف، بمساعدة شريط الإزاحة الأفقي.

إنشاء برنامج الوسائط المتعددة

لنعمل الآن على إنشاء برنامج الوسائط المتعددة:

- أنشئ الدليل C:\VB5Prg\Ch19، لكي تحفظ عملك فيه.
- ابدأ مشروعاً جديداً بنوع Standard EXE.
- احفظ النموذج From1 باسم MCI.frm، في الدليل C:\VB5Prg\Ch19. واحفظ المشروع باسم MCI.Vbp في نفس الدليل السابق.

إضافة عنصر تحكم OCX إلى المشروع

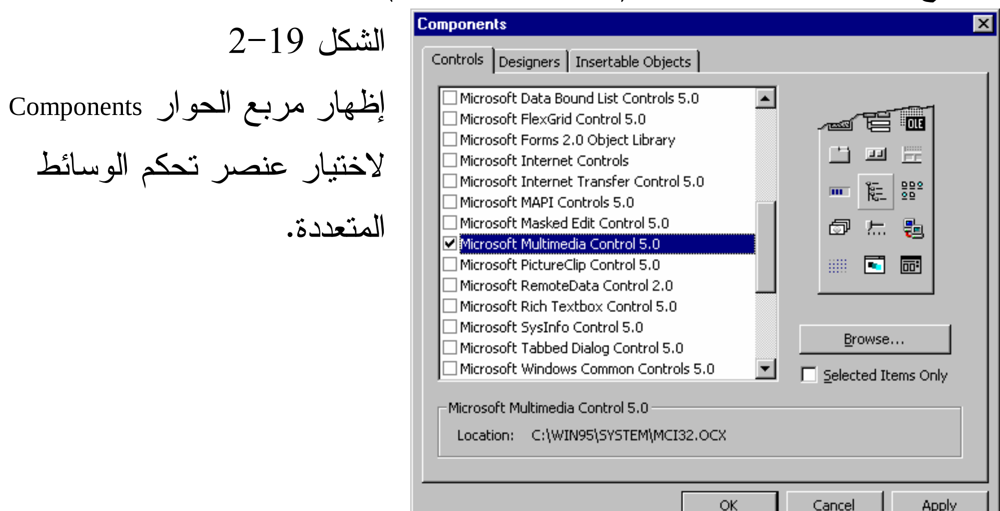
قبل التمكن من وضع عنصر OCX على النموذج، يجب أولاً إضافته إلى المشروع نفسه.

سنضيف العنصر MCI32.OCX إلى مشروعنا MCI.Vbp:

- اختر البند Component من قائمة **Project**.

يستجيب فيجول بيسك بإظهار مربع الحوار Component.

- اختر البند Microsoft Multimedia Control 5.0 من صفحة Controls في مربع الحوار Components (انظر الشكل 19−2).



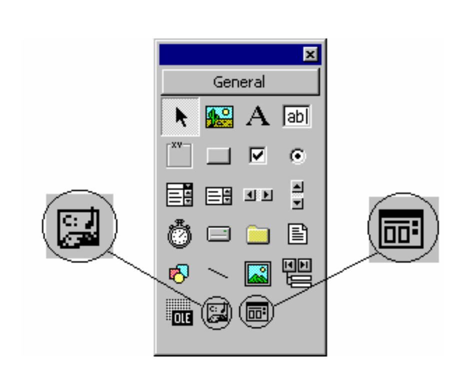
الشكل 19−2

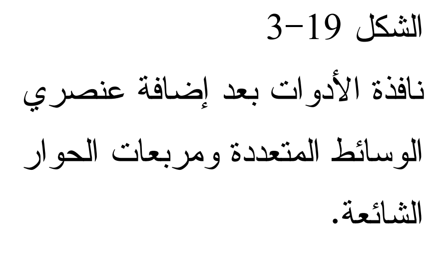
إظهار مربع الحوار Components لاختيار عنصر تحكم الوسائط المتعددة.

- قبل الخروج من مربع الحوار Components، اختر البند Microsoft Common Dialog Control 5.0 أيضاً، لأننا سنحتاجه في برنامج الوسائط المتعددة.

- انقر الزر **OK** للخروج من مربع الحوار Components.

بعد تنفيذ الخطوات السابقة، ينبغي ظهور رمزين جديدين: أحدهما لعنصر تحكم الوسائط المتعددة والثاني لعنصر تحكم مربعات الحوار الشائعة. (انظر الشكل 19−3).





الشكل 19-3
نافذة الأدوات بعد إضافة عنصري الوسائط المتعددة ومربعات الحوار الشائعة.

التصميم المرئي لبرنامج الوسائط المتعددة

لنبدأ الآن عملية التصميم المرئي لبرنامج الوسائط المتعددة:

- أسند القيم الموضحة لاحقاً، للخصائص التي تتبع لها للنموذج الافتراضي Form1:

```
Name:          frmMCI
Caption:       برنامج الوسائط المتعددة
RightToLeft:   True
```

اربط القائمة الموضحة بالجدول 19-1 مع النموذج frmMCI.

جدول 19-1. قائمة النموذج frmMCI.

Name	Caption
mnuFile	&ملف
mnuOpenWave	...فتح ملف &صوت
mnuOpenAVI	...فتح ملف &حركة
mnuOpenMIDI	...فتح ملف &موسيقى
mnuSep1	-...
mnuExit	**...&خروج**

- أضف عنصر الوسائط المتعددة للنموذج frmMCI، وأسند القيمة mciMyMedia للخاصية Name.
- أضف عنصر مربعات الحوار الشائعة للنموذج frmMCI، وأسند القيمة dlgGetFile للخاصية Name.

بعد الانتهاء من الخطوات السابقة، ينبغي أن يصبح النموذج frmMCI، كما في الشكل 19-4.



الشكل 19-4

النموذج frmMCI بعد إضافة القائمة وعنصري الوسائط المتعددة ومربعات الحوار الشائعة.

لنكمل الآن عملية بناء النموذج frmMCI:

- أضف عنصر تحكم الوقت Timer للنموذج frmMCI.
- أسند له الخصائص التالية:

```
Name:          tmrGetPosition
Interval:      50
Enabled:       False
```

- أضف باقي العناصر وفقاً للجدول 19-2 (انتبه لأنك أضفت بعض العناصر الموجودة في الجدول 19-2).

جدول 19-2 العناصر المكونة لبرنامج الوسائط المتعددة وخصائصها.

الكائن	الخاصية	القيمة
Form	**Name**	**frmMCI**
	Caption	برنامج الوسائط المتعددة
	RightToLeft	True
MMControl	**Name**	**mciMyMedia**
Common Dialy	**Name**	**dlgGetFile**
Timer	**Name**	**tmrGetPosition**
	Interval	50
	Enabled	False
Label	**Name**	**lblTitle1**
	Caption	اسم الملف الحالي

القيمة	الخاصية	الكائن
lblTitle2	**Name**	**Label**
الموقع الحالي	Caption	
lblTitle3	**Name**	**Label**
طول الملف	Caption	
lblCurrentFile	**Name**	**Label**
فارغ	Caption	
lblCurPosition	**Name**	**Label**
فارغ	Caption	
lblLastPosition	**Name**	**Label**
فارغ	Caption	
hsbPosition	**Name**	**HscrollBar**

بعد الانتهاء من إضافة جميع العناصر الموجودة في الجدول السابق، ينبغي أن يظهر النموذج frmMCI كما في الشكل 19-5.



الشكل 19-5

النموذج frmMCI بعد الانتهاء من عملية تصميمه.

كتابة نص برنامج الوسائط المتعددة

بعد الانتهاء من تصميم النموذج frmMCI، علينا كتابة نص البرنامج:

- اكتب الأسطر التالية في الإجراء ()mnuOpenWave_Click:

```
Private Sub mnuOpenWave_Click()
    Dim mmFileName As String
    'filename تهيئة الخاصية
    dlgGetFile.filename = ""
    'تحديد نوع الملف الذي سيفتحه مربع الحوار
    dlgGetFile.Filter = "ملفات الصوت(*.wav)|*.wav"
```

```
    'إظهار مربع الحوار فتح ملف
    dlgGetFile.ShowOpen
    'mmFileName إسناد الملف الذي اختاره المستخدم للمتحول
    mmFileName = dlgGetFile.filename
    'إذا لم يختر المستخدم ملفا، فاخرج من الإجراء فورا
    If mmFileName = "" Then
      Exit Sub
    End If
    'إغلاق الجهاز السابق في حال وجود جهاز مفتوح
    mciMyMedia.Command = "Close"
    'تحديد نوع الجهاز بأنه جهاز ملفات الصوت
    mciMyMedia.DeviceType = "WaveAudio"
    'تحديد الملف المراد فتحه
    mciMyMedia.filename = mmFileName
    'إعطاء أمر فتح للجهاز الصوتي
    mciMyMedia.Command = "Open"
    'تحديد أكبر موقع ممكن لشريط الإزاحة الأفقي
    hsbPosition.Max = mciMyMedia.Length
    'عرض قيمة طول الملف في أداة العنونة
    lblLastPositiom.Caption = mciMyMedia.Length
    'عرض الملف الحالي في أداة العنونة
    lblCurrentFile.Caption = mmFileName
End Sub
```

- اكتب الأسطر التالية في الإجراء ()mnuOpenAVI_Click:

```
Private Sub mnuOpenAVI_Click()
    Dim mmFileName As String

    'filename تهيئة الخاصية
    dlgGetFile.filename = ""
    'تحديد نوع الملف الذي سيفتحه مربع الحوار
    dlgGetFile.Filter = "ملفات الحركة(*.avi)|*.avi"
    'إظهار مربع الحوار فتح ملف
    dlgGetFile.ShowOpen
    'mmFileName إسناد الملف الذي اختاره المستخدم للمتحول
    mmFileName = dlgGetFile.filename
    'إذا لم يختر المستخدم ملفا، فاخرج من الإجراء فورا
    If mmFileName = "" Then
```

```
  Exit Sub
End If

'إغلاق الجهاز السابق في حال وجود جهاز مفتوح
mciMyMedia.Command = "Close"
'تحديد نوع الجهاز بأنه جهاز ملفات الحركة
mciMyMedia.DeviceType = "AVIVideo"
'تحديد الملف المراد فتحه
mciMyMedia.filename = mmFileName
'إعطاء أمر فتح للجهاز الحركي
mciMyMedia.Command = "Open"

'تحديد أكبر موقع ممكن لشريط الإزاحة الأفقي
hsbPosition.Max = mciMyMedia.Length
'عرض قيمة طول الملف في أداة العنونة
lblLastPositiom.Caption = mciMyMedia.Length
'عرض الملف الحالي في أداة العنونة
lblCurrentFile.Caption = mmFileName
End Sub
```

- اكتب الأسطر التالية في الإجراء mnuOpenMIDI_Click():

```
Private Sub mnuOpenMIDI_Click()
  Dim mmFileName As String

  'تهيئة الخاصية filename
  dlgGetFile.filename = ""
  'تحديد نوع الملف الذي سيفتحه مربع الحوار
  'dlgGetFile.Filter = "All Files|(*.rmi)"
  'إظهار مربع الحوار فتح ملف
  dlgGetFile.ShowOpen
  'إسناد الملف الذي اختاره المستخدم للمتحول mmFileName
  mmFileName = dlgGetFile.filename
  'إذا لم يختر المستخدم ملفا، فاخرج من الإجراء فورا
  If mmFileName = "" Then
    Exit Sub
  End If

  'إغلاق الجهاز السابق في حال وجود جهاز مفتوح
```

```
    mciMyMedia.Command = "Close"
    'تحديد نوع الجهاز بأنه جهاز ملفات الموسيقى
    mciMyMedia.DeviceType = "Sequencer"
    'تحديد الملف المراد فتحه
    mciMyMedia.filename = mmFileName
    'إعطاء أمر فتح للجهاز الصوتي
    mciMyMedia.Command = "Open"
    'تحديد أكبر موقع ممكن لشريط الإزاحة الأفقي
    hsbPosition.Max = mciMyMedia.Length
    'عرض قيمة طول الملف في أداة العنونة
    lblLastPositiom.Caption = mciMyMedia.Length
    'عرض الملف الحالي في أداة العنونة
    lblCurrentFile.Caption = mmFileName
End Sub
```

- اكتب الأسطر التالية في الإجراء ()mnuExit_Click:

```
Private Sub mnuExit_Click()
    End
End Sub
```

- اكتب الأسطر التالية في الإجراء ()trmGetPosition_Timer:

```
Private Sub tmrGetPosition_Timer()
    lblCurPosition.Caption = mciMyMedia.Position
    hsbPosition.Value = mciMyMedia.Position
End Sub
```

- اكتب الأسطر التالية في الإجراء ()mciMyMedia_PlayClick:

```
Private Sub mciMyMedia_Click()

End Sub
```

- اكتب الأسطر التالية في الإجراء ()Form_Unload:

```
Private Sub Form_Unload(Cancel As Integer)
    mciMyMedia.Command = "Close"
End Sub
```

- احفظ عملك الآن.

تنفيذ برنامج الوسائط المتعددة

بعد الانتهاء من كتابة نص البرنامج، يصبح جاهزاً للتنفيذ:

- نفّذ البرنامج بضغط المفتاح F5.

يستجيب فيجول بيسك بإظهار نافذة برنامج الوسائط المتعددة.

- اختر البند **فتح ملف صوتي** من قائمة ملف.

يستجيب برنامج الوسائط المتعددة بإظهار مربع الحوار فتح.

- اختر أحد الملفات الصوتية التي لها الامتداد Wav، ثم انقر الزر **فتح** في مربع الحوار فتح.

تجد أمثلة عن ملفات الصوت في القرص المدمج المرفق مع الكتاب، في دليل الوسائط المتعددة.

- يعرض برنامج الوسائط المتعددة اسم الملف الذي اخترته ومساره الكامل، كما يعرض أيضاً طول الملف بوحدة جزء من الألف من الثانية.

استعد برنامج الوسائط المتعددة لعزف الملف الذي اخترته:

- انقر الزر **Play** الموجود على عنصر تحكم الوسائط المتعددة.

يستجيب برنامج الوسائط المتعددة بعزف الملف الذي اخترته، ويظهر لك الموقع الحالي للعزف بطريقتين:

  - رقم يحدد الزمن الذي تم عزفه إلى الآن.
  - استخدام مؤشر شريط الإزاحة الأفقي.

- لإعادة عزف الملف مرة ثانية، يجب نقر الزر **Back** (السهم الموجود فى أقصى اليسار)، لإعادة مؤشر العزف إلى بداية الملف مرة أخرى، ثم نقر الزر **Play**.

- أثناء عزف الملف، تستطيع نقر الزر **Stop** (مربع غامق)، لإيقاف العزف نهائياً، أو زر **Pause** (خطين شاقوليين) لإيقاف العزف مؤقتاً.

- لعزف نوع آخر من الملفات وليكن ملفات الحركة، اختر البند **فتح ملف حركة** من قائمة **ملف**، واختر أحد الملفات التي لها الامتداد AVI (ملفات حركة).

تجد أمثلة عن ملفات الحركة، في القرص المدمج المرفق مع الكتاب، في دليل الوسائط المتعددة.

يعرض البرنامج أيضاً معلومات عن الملف الذي اخترته مثل اسم الملف ومساره الكامل، وطول الملف بالثواني.

- انقر الزر **Play** لعرض الملف الذي اخترته.

يستجيب برنامج الوسائط المتعددة بفتح نافذة، يعرض الفيلم (ملف AVI) عبرها. وعنوان النافذة عبارة عن اسم الملف الذي يتم عرضه فيها.

- انقر الزر **Next Frame** أو الزر **Previous Frame**، لعرض الملف صورة صورة (Frame by Frame).
- تستطيع إغلاق نافذة عرض الفيلم بنقر أيقونة الإغلاق في النافذة نفسها، أو ببساطة فتح ملف آخر (مهما كان نوعه). لأن فتح ملف جديد يتسبب في إغلاق الملف الحالي آلياً.

لفتح النوع الثالث من ملفات الوسائط المتعددة:

- اختر البند **فتح ملف موسيقي** من قائمة **ملف**، واختر أحد الملفات التي لها الامتداد MID.

تجد أمثلة عن ملفات الموسيقى في القرص المدمج المرفق مع الكتاب، في دليل الوسائط المتعددة.

- انقر الزر **Play** لعزف الملف الموسيقي الذي اخترته.
- تمرن على البرنامج بجميع أقسامه ثم اختر البند **خروج** من قائمة **ملف** لإنهاء البرنامج.

كيف يعمل برنامج الوسائط المتعددة

يعتمد برنامج الوسائط المتعددة اعتماداً أساسياً على عنصر تحكم الوسائط المتعددة mciMyMedia. وتنحصر وظيفة البرنامج في تحديد الخصائص، وإعطاء الأوامر لهذا العنصر، ليقوم بفتح وعزف ملفات الوسائط المتعددة المختلفة.

نص البرنامج للإجراء mnuOpenWave_Click()

ينفذ هذا الإجراء عند اختيار المستخدم البند **فتح ملف صوتي** من قائمة **ملف** في برنامج الوسائط المتعددة. يسمح هذا الإجراء للمستخدم، باختيار ملف صوتي (WAV)، ومن ثم فتحه وعزفه.

لقد كتبت النص التالي في الإجراء mnuOpenWave_Click():

```
Private Sub mnuOpenWave_Click()
    Dim mmFileName As String

    'filename تهيئة الخاصية
    dlgGetFile.filename = ""
    'تحديد نوع الملف الذي سيفتحه مربع الحوار
    dlgGetFile.Filter = "ملفات الصوت(*.wav)|*.wav"
    'إظهار مربع الحوار فتح ملف
    dlgGetFile.ShowOpen
    'mmFileName إسناد الملف الذي اختاره المستخدم للمتحول
    mmFileName = dlgGetFile.filename
    'إذا لم يختر المستخدم ملفا، فاخرج من الإجراء فورا
    If mmFileName = "" Then
      Exit Sub
    End If
    'إغلاق الجهاز السابق في حال وجود جهاز مفتوح
    mciMyMedia.Command = "Close"
    'تحديد نوع الجهاز بأنه جهاز ملفات الصوت
    mciMyMedia.DeviceType = "WaveAudio"
    'تحديد الملف المراد فتحه
    mciMyMedia.filename = mmFileName
    'إعطاء أمر فتح للجهاز الصوتي
    mciMyMedia.Command = "Open"
    'تحديد أكبر موقع ممكن لشريط الإزاحة الأفقي
    hsbPosition.Max = mciMyMedia.Length
    'عرض قيمة طول الملف في أداة العنونة
    lblLastPositiom.Caption = mciMyMedia.Length
    'عرض الملف الحالي في أداة العنونة
    lblCurrentFile.Caption = mmFileName
```

```
End Sub
```

صرحت العبارة الأولى عن المتحول mmFileName، ووظيفته تخزين اسم الملف الذي اختاره المستخدم. أما العبارة:

```
dlgGetFile.filename = ""
```

تهيئ الخاصية FileName التابعة لمربع الحوار dlgGetFile وإسناد القيمة "لا شيء" لها.

بعد ذلك، جاء دور معرفة نوع الملف الذي سيفتحه المستخدم (معرفة امتداد الملف) عن طريق العبارة التالية:

```
dlgGetFile.Filter = "ملفات الصوت(*.wav)|*.wav"
```

يتم إسناد أنواع الملفات التي يستطيع مربع الحوار فتحها، إلى الخاصية Filter التابعة لمربع الحوار dlgGetFile.

بعد معرفة نوع الملفات التي نريد فتحها، نعطي الأمر بالظهور لمربع الحوار فتح، عن طريق الأمر ShowOpen وذلك كالتالي:

```
dlgGetFile.ShowOpen
```

تتوقف عملية التنفيذ عند هذا السطر، لحين انتهاء المستخدم من اختيار ملف ما، ونقر الزر **فتح** الموجود في مربع الحوار.

هنا لدينا احتمالان:

- اختار المستخدم ملفاً صوتياً.
- لم يختر المستخدم أي ملف، أو نقر الزر **إلغاء الأمر** في مربع الحوار.

لمعرفة نوع الاحتمال الحاصل، لجأنا إلى فحص اسم الملف الذي اختاره المستخدم، عن طريق العبارة If ... Then التالية:

```
If mmFileName = "" Then
  Exit Sub
End If
```

يفحص هذا الشرط قيمة المتحول mmFileName، الممثل لاسم الملف الذي اختاره المستخدم، إذا كانت قيمته تساوي "لا شيء"، (لم يختر المستخدم ملفاً)، فإنه ببساطة يخرج بدون إكمال تنفيذ باقي الإجراء.

لا بد أن تكون لاحظت الآن، لماذا قمنا بتهيئة الخاصية FileName التابعة لمربع الحوار، وإسناد القيمة "لا شيء" لها، قبل إظهار مربع الحوار. لأننا سنفحص هذه الخاصية بعد الخروج من مربع الحوار، ونرى هل أصبح لها قيمة أخرى غير قيمة "لا شيء"، والتي تعني أن المستخدم لم يختر ملفاً كما ذكرنا سابقاً.

ينفذ باقي الإجراء، عند اختيار ملف صوتي، ويبدأ من العبارة التالية:

```
mciMyMedia.Command = "Close"
```

تُغلق هذه العبارة، أي جهاز مفتوح مسبقاً، (يقصد بالجهاز هنا: أحد ملفات الوسائط المتعددة المفتوحة والجاهزة للعزف).

بعد إغلاق الجهاز المفتوح (في حال وجوده)، يُحدد نوع الجهاز المطلوب فتحه أو تشغيله (وهو في هذه الحالة، الجهاز "Wave Audio"). يحدد نوع الجهاز حسب نوع الملف المطلوب فتحه.

إذا كان نوع الملف صوتياً، يحدد الجهاز على أنه من النوع "Wave Audio"، أما في حال كونه موسيقياً، يكون نوع الجهاز الصحيح هو "Sequencer"، وفي حال كونه نوع الجهاز فيلماً متحركاً، يكون نوع الجهاز "AVI Video" .

نُسند نوع الجهاز المطابق، للخاصية DeviceType التابعة لعنصر الوسائط المتعددة كالتالي:

```
mciMyMedia.DeviceType = "WaveAudio"
```

بعد تحديد نوع الجهاز المطلوب، نسند اسم الملف الذي اختاره المستخدم إلى الخاصية FileName التابعة لعنصر الوسائط المتعددة mciMyMedia، حتى يتعرف عنصر الوسائط المتعددة على اسم ومسار الملف المراد فتحه.

بعد معرفة اسم الملف، نُعطى لعنصر الوسائط المتعددة، الأمر Open عن طريق العبارة التالية:

```
mciMyMedia.Command = "Open"
```

ليعمل على فتح الملف وتهيئته للعزف، ومعرفة كل المعلومات الخاصة بهذا الملف.

أحد هذه المعلومات، هي طول الملف الصوتي، (بوحدة الملي ثانية)، حيث نستطيع الحصول عليها عن طريق الخاصية Length التابعة لعنصر الوسائط المتعددة mciMyMedia.

استفدنا من قيمة طول الملف الصوتي، في تحديد أكبر مجال ممكن لشريط الإزاحة الأفقي، عن طريق إسناد هذه القيمة للخاصية Max التابعة لشريط الإزاحة الأفقي كالتالي:

```
hsbPosition.Max = mciMyMedia.Length
```

أظهرنا قيمة طول الملف الصوتي في اللافتة، حتى يعرف المستخدم طول الملف المفتوح، بأجزاء من الألف من الثانية، وذلك كالتالي:

```
lblLastPosition.Caption = mciMyMedia.Length
```

أيضاً، أظهرنا اسم الملف ومساره الكامل، (حتى يعرف المستخدم اسم الملف المفتوح الحالي) في لافتة أخرى، عن طريق العبارة التالية:

```
lblCurrentFile.Caption = mmFileName
```

بعد الانتهاء من تنفيذ كامل عبارات هذا الإجراء، يصبح الملف الذي اختاره المستخدم جاهزاً للعزف.

نص البرنامج للإجراء mnuOpenMIDI_Click ()

ينفذ هذا الإجراء عند اختيار المستخدم البند **فتح ملف موسيقى** من قائمة ملف:

```
Private Sub mnuOpenMIDI_Click()
    Dim mmFileName As String
    'filename تهيئة الخاصية
    dlgGetFile.filename = ""
    'تحديد نوع الملف الذي سيفتحه مربع الحوار
    dlgGetFile.Filter = "ملفات الموسيقى(*.mid)|*.mid"
    'إظهار مربع الحوار فتح ملف
    dlgGetFile.ShowOpen
    'mmFileName إسناد الملف الذي اختاره المستخدم للمتحول
    mmFileName = dlgGetFile.filename
    'إذا لم يختر المستخدم ملفا، فاخرج من الإجراء فورا
    If mmFileName = "" Then
      Exit Sub
```

```
    End If
    'إغلاق الجهاز السابق في حال وجود جهاز مفتوح
    mciMyMedia.Command = "Close"
    'تحديد نوع الجهاز بأنه جهاز ملفات الموسيقى
    mciMyMedia.DeviceType = "Sequencer"
    'تحديد الملف المراد فتحه
    mciMyMedia.filename = mmFileName
    'إعطاء أمر فتح للجهاز الصوتي
    mciMyMedia.Command = "Open"
    'تحديد أكبر موقع ممكن لشريط الإزاحة الأفقي
    hsbPosition.Max = mciMyMedia.Length
    'عرض قيمة طول الملف في أداة العنونة
    lblLastPositiom.Caption = mciMyMedia.Length
    'عرض الملف الحالي في أداة العنونة
    lblCurrentFile.Caption = mmFileName
End Sub
```

يعمل هذا الإجراء، بنفس الأسلوب الذي شرحناه سابقاً مع وجود اختلافين اثنين هما:

- تحديد نوع الملف الذي يستطيع المستخدم اختياره عن طريق مربع الحوار، على أنه النوع ذو الامتداد (MID). والعبارة المسئولة عن ذلك هي:

```
    dlgGetFile.Filter = "ملفات الموسيقى(*.mid)|*.mid"
```

- تحديد نوع جهاز آخر يمكنه فتح الملفات الموسيقية، وهو الجهاز ذو النوع "Sequencer"، عن طريق العبارة التالية:

```
    mciMyMedia.DeviceType = "Sequencer"
```

نص البرنامج للإجراء mnuOpenAVI_Click()

- ينفذ هذا الإجراء آلياً، عند اختيار البند **فتح ملف حركة** من قائمة ملف:

```
Private Sub mnuOpenAVI_Click()
    Dim mmFileName As String

    'filename تهيئة الخاصية
    dlgGetFile.filename = ""
    'تحديد نوع الملف الذي سيفتحه مربع الحوار
    dlgGetFile.Filter = "ملفات الحركة(*.avi)|*.avi"
    'إظهار مربع الحوار فتح ملف
```

```
    dlgGetFile.ShowOpen
    'mmFileName إسناد الملف الذي اختاره المستخدم للمتحول
    mmFileName = dlgGetFile.filename
    'إذا لم يختر المستخدم ملفا، فاخرج من الإجراء فورا
    If mmFileName = "" Then
      Exit Sub
    End If

    'إغلاق الجهاز السابق في حال وجود جهاز مفتوح
    mciMyMedia.Command = "Close"
    'تحديد نوع الجهاز بأنه جهاز ملفات الحركة
    mciMyMedia.DeviceType = "AVIVideo"
    'تحديد الملف المراد فتحه
    mciMyMedia.filename = mmFileName
    'إعطاء أمر فتح للجهاز الحركي
    mciMyMedia.Command = "Open"

    'تحديد أكبر موقع ممكن لشريط الإزاحة الأفقي
    hsbPosition.Max = mciMyMedia.Length
    'عرض قيمة طول الملف في أداة العنونة
    lblLastPositiom.Caption = mciMyMedia.Length
    'عرض الملف الحالي في أداة العنونة
    lblCurrentFile.Caption = mmFileName
End Sub
```

- يعمل هذا الإجراء أيضاً، بنفس أسلوب عمل الإجراءين السابقين، ومع وجود نفس الاختلافين السابقين وهما:
  - تحديد نوع الملفات التي نريد اختيارها من مربع الحوار، على أنها من النوع AVI، وذلك كالتالي:

```
dlgGetFile.Filter = "ملفات الحركة(*.avi)|*.avi"
```

  - تحديد نوع الجهاز القادر على عرض الملف الحركي، وهو الجهاز ذو النوع "AVIVideo"، عن طريق العبارة التالية:

```
mciMyMedia.DeviceType = "AVIVideo"
```

ملاحظة

يتطلب برنامج الوسائط المتعددة وجود بطاقة صوت، لإصدار الأصوات والموسيقى من خلاله، أما إظهار الملفات الحركية، فهي لا تتطلب بطاقات إضافية.

التقنية المستخدمة لعرض الموقع الحالي خلال عزف الملف

بعد اختيار أحد ملفات الوسائط المتعددة، وفتحه عن طريق القائمة **ملف**، يصبح البرنامج جاهزاً لعزف هذا الملف، عن طريق النقر على الزر **Play**، يظهر لك الموقع الحالي للعزف خلال عزفه، بأسلوبين:

- إظهار الموقع الحالي للعزف، كرقم يمثل الوقت الذي انقضى منه، في اللافتة lblCurPosition.
- إظهار الموقع الحالي للعزف، عن طريق مؤشر شريط الإزاحة الأفقي hsbPosition.

كيف يعلم البرنامج الموقع الحالي للعزف ويظهره للمستخدم؟

نستطيع معرفة الموقع الحالي للعزف، عن طريق قراءة قيمة الخاصية Position التابعة لعنصر الوسائط المتعددة mciMyMedia.

لكننا نريد معرفة الموقع الحالي للعزف، خلال مرحلة العزف نفسها، وعلى فترات صغيرة جداً. لذلك استخدمنا لهذه الغاية، عنصر تحكم الميقاتية Timer، وجعلناه يقرأ الموقع الحالي للعزف، وإظهاره كل 20/1 من الثانية.

أسندنا القيمة 50 للخاصية Interval التابعة للميقاتية trmGetPosition، خلال مرحلة التصميم، وتعني تنفيذ الحادثة Timer التابعة للميقاتية، كل 50 ميلي ثانية.

نص البرنامج المكتوب في الحادثة Timer هو:

```
Private Sub tmrGetPosition_Timer()
    lblCurPosition.Caption = mciMyMedia.Position
    hsbPosition.Value = mciMyMedia.Position
End Sub
```

كما قلنا سابقاً، يُنفذ هذا الإجراء آلياً كل 50 ميلي ثانية، ويُظهر قيمة الخاصية Position التابعة لعنصر الوسائط المتعددة (الممثلة لموقع العزف الحالي)، في اللافتة lblCurPosition كالتالي:

```
lblCurPosition.Caption = mciMyMedia
```

يُسند أيضاً قيمة الخاصية Position التابعة لعنصر الوسائط المتعددة، للخاصية Value التابعة لشريط الإزاحة الأفقي، لتنقل مؤشر شريط الإزاحة الأفقي للموقع الحالي للعزف، كالتالي:

```
hsbPosition.Valve = mciMyMedia.Position
```

وبذلك نضمن تحديث دائم، لموقع العزف الحالي الظاهر للمستخدم.

لكن توجد مشكلة مع التوقيت، هل عرفتها؟.

يعمل التوقيت دائماً خلال مرحلة عزف الملف، وخلال فترة التوقف أيضاً، وهذا الأمر غير مقبول برمجياً (عمل التوقيت الدائم، خلال فترة تنفيذ البرنامج).

إذاً، يجب تشغيل الميقاتية خلال فترة عزف الملف فقط، وإيقافه في الفترات الأخرى، وذلك عن طريق الخاصية Enabled، التي تعطّله إذا أُسندت القيمة False إليه، أو تجعله يعمل إذا أُسندت القيمة True إليها.

أسندنا القيمة False للخاصية Enabled التابعة للتوقيت، خلال مرحلة تصميم البرنامج، لأننا نريد تعطيل التوقيت عند بداية تشغيل البرنامج. لذلك، يجب تفعيل التوقيت فقط، خلال فترة عزف الملف، وتعطيلها خارج هذه الفترة.

يوفّر عنصر تحكم الوسائط المتعددة، الحادثة Play_Click()، التي تنفذ آلياً عند نقر المستخدم للزر **Play** (بدء العزف)، وهي أنسب مكان لوضع العبارة التي تفعّل التوقيت، وذلك كالتالي:

```
Private Sub mciMyMedia_PlayClick(Cancel As Integer)
    tmrGetPosition.Enabled = True
End Sub
```

بعد الانتهاء من العزف، ينفذ عنصر تحكم الوسائط المتعددة الحادثة Done (نهاية العزف) آلياً. لذلك، هذه الحادثة تعتبر الموقع المناسب لوضع العبارة التي تعطّل

التوقيت مرة ثانية، (لأن العزف انتهى، ولا داعي لبقاء التوقيت تعمل في هذه الفترة). وذلك كالتالي:

```
Private Sub mciMyMedia_Done(NotifyCode As Integer)
    tmrGetPosition.Enabled = False
End Sub
```

نص البرنامج للإجراء () Form_Unload

ينفذ الإجراء Form_Unload() آلياً، عند إنهاء البرنامج وإزالته من ذاكرة ويندوز. كما أن الحادثة Form_Load() تنفذ آلياً، عند بداية تشغيل البرنامج ووضع النافذة في الذاكرة. لضمان إغلاق كافة الأجهزة المفتوحة من قبل عنصر تحكم الوسائط المتعددة، وضعنا العبارة التالية في الإجراء Form_Unload() كالتالي:

```
Private Sub Form_Unload(Cancel As Integer)
    mciMyMedia.Command = "Close"
End Sub
```

تطوير برنامج الوسائط المتعددة

سنطور الآن برنامج الوسائط المتعددة، ليصبح قادراً على قراءة الأقراص الليزرية الموسيقية.

- أضف بنداً جديداً للقائمة **ملف**، باسم mnuOpenCDAudio، وعنوان **فتح قرص ليزري موسيقي**.

بعد إضافة البند الجديد، ينبغي أن يصبح البرنامج كما في الشكل 6-19.



الشكل 6-19

برنامج الوسائط المتعددة بعد إضافة البند **فتح قرص ليزري موسيقى** له.

- اكتب الأسطر التالية في الإجراء mnuOpenCDAudio_Click():

```
Private Sub mnuOpenCDAudio_Click()
```

```
    'إغلاق الجهاز السابق في حال وجود جهاز مفتوح
    mciMyMedia.Command = "Close"
    'تحديد نوع الجهاز بأنه جهاز القرص الموسيقي
    mciMyMedia.DeviceType = "CDAudio"
    'إعطاء أمر الفتح لجهاز القرص الليزري الموسيقي
    mciMyMedia.Command = "Open"
    hsbPosition.Max = mciMyMedia.Length
    'عرض قيمة طول المسار في أداة العنونة
    lblLastPositiom.Caption = mciMyMedia.Length
    lblCurrentFile.Caption = "فتح القرص الليزري الموسيقي"
End Sub
```

لا يعتبر القرص الليزري الموسيقي ملفاً، لذلك لا حاجة لوجود مربع حوار اختيار ملف.

ينفذ هذا الإجراء آلياً عند اختيار المستخدم البند **فتح قرص ليزري موسيقي** من قائمة **ملف**.

يُغلق الجهاز السابق أولاً، ثم يُفتح الجهاز "CDAudio" الخاص بالأقراص الليزرية الموسيقية، كما هو واضح من عبارات الإجراء السابق.

ملاحظة

لكي تستطيع تشغيل هذا الجزء من البرنامج، يجب تواجد قرص ليزري موسيقي لديك. وأن يكون مشغل الأقراص الليزرية موصول بسلك مباشر مع بطاقة الصوت. أو يمكنك سماع الأقراص الليزرية الموسيقية، من زر سماعة الرأس الموجود في واجهة مشغل الأقراص الليزرية.

- بعد فتح الجهاز الليزري الموسيقي، يصبح عزف المسار الأول من القرص الليزري الموسيقي ممكناً، للانتقال إلى المسار اللاحق من القرص، انقر الزر NextTrack (هو نفس الزر الذي ينقلنا لنهاية الملف). للانتقال إلى المسار السابق،
- انقر الزر PreviousTrack (هو نفس الزر الذي ينقلنا لبداية الملف).
- انقر الزر **Eject** لفتح علبة مشغل الأقراص الليزرية.

لاحظ أن الأزرار التابعة لعنصر تحكم الوسائط المتعددة، تختلف معانيها تقريباً، حسب

نوع الجهاز المفتوح.

- أضف الآن الأسطر التالية للإجراء ()mciMyMedia_NextClick:

```
Private Sub mciMyMedia_NextClick(Cancel As Integer)
    hsbPosition.Max = mciMyMedia.Length
    lblLastPositiom.Caption = mciMyMedia.Length
End Sub
```

ينفذ هذا الإجراء آلياً، عند نقر الزر **NextTrack** (مسار لاحق)، حيث يقوم بإظهار طول المسار اللاحق الذي تم الانتقال إليه.

- أضف الآن الأسطر التالية للإجراء ()mciMyMedia_PrevClick:

```
Private Sub mciMyMedia_PrevClick(Cancel As Integer)
    hsbPosition.Max = mciMyMedia.Length
    lblLastPositiom.Caption = mciMyMedia.Length
End Sub
```

ينفذ هذا الإجراء آلياً، عند نقر الزر PrevTrack (مسار لاحق)، وله نفس وظيفة الإجراء السابق.

لمحة عن أنواع ملفات الوسائط المتعددة

سنشرح الآن بعض أهم أنواع ملفات الوسائط المتعددة، مع ذكر أهم الفروقات بين هذه الأنواع، وسبب وجود هذه الأنواع أصلاً:

الملفات الصوتية Wave

تملك هذه الملفات الامتداد (wav)، ويخزن الصوت فيها بشكل رقمي Digital.تستطيع هذه الملفات، تخزين جميع أنواع الأصوات، مثل الموسيقى والغناء والكلام البشري أيضاً. بكلام آخر، يمكن اعتبار ملف الصوت wav، مثل جهاز التسجيل المنزلي Recorder.

يمكن تخزين الصوت في هذه الملفات على عدة تنسيقات، أهمها:

صوت 8 كيلو بايت/الثانية	8 بت	أحادي	8000 هرتز
صوت 16 كيلو بايت/الثانية	8 بت	استريو	8000 هرتز

صوت 43 كيلو بايت/الثانية	16 بت	أحادي	22050 هرتز
صوت 86 كيلو بايت/الثانية	16 بت	استريو	22050 هرتز
صوت 86 كيلو بايت/الثانية	16 بت	أحادي	22050 هرتز
صوت 172 كيلو بايت/الثانية	16 بت	استريو	44100 هرتز

يعتبر تخزين الصوت بتنسيق أحادي 8000 هرتز، هو أدنى جودة ممكنة، وهي مشابهة لجودة صوت سماعة التلفون.

أما تخزين الصوت بتنسيق 22050 هرتز، فهي جودة متوسطة تشابه جودة صوت الراديو.

وتخزين الصوت بتنسيق 44100 هرتز، فهي جودة عالية جداً تشابه جودة صوت القرص الليزري الموسيقي.

يأتي تخزين الصوت بجودة عالية، على حساب حجم الملف، فمثلاً، تخزين خمسة دقائق من الصوت بجودة عالية جداً وصوت ستيريو، يحتاج لحجم مقداره 172 × 5 × 60 = 51600 كيلو بايت، أو 51.6 ميجا بايت.

ولتخزين نفس الملف بجودة متوسطة مثلاً، وصوت أحادي، يحتاج لحجم مقداره 43 × 5 × 60 = 12900 كيلو بايت أو 12.9 ميغا بايت. وهكذا.

إذاً، توصلنا لنتيجة مفادها: كلما زادت جودة الصوت المسجل، كلما زاد حجم الملف الذي نسجل فيه الصوت.

الملفات الموسيقية MIDI

تصور أنك تريد تخزين الموسيقى أو النوتات الموسيقية في ملف ما، فليس من المعقول أبداً تخزين الموسيقى في ملف Wave، لأنه في هذه الحالة، سيأخذ حجماً كبيراً جداً، لذلك يمكنك تخزين النوتات الموسيقية، مع جميع الأوامر الموسيقية الأخرى، داخل الملف من النوع MID.

يكون حجم الملف MID صغيراً جداً، مقارنة مع أحجام الملفات Wave.

عند عزف الملف الموسيقي MID، يتعرف ويندوز على نوع النوتة الموسيقية التي يجب عزفها، ويرسلها كأمر إلى بطاقة الصوت (أو جهاز الأورج مثلاً)، ليصدر الصوت الخاص بهذه النوتة.

نستطيع تخيل الملف الموسيقي MID، على أنه تسجيل لضربات المفاتيح التي يقوم بها العازف على الأورج.

لا بد أنك توصلت إلى نتيجة أنه لا يمكنك تخزين صوت آخر غير النوتات الموسيقية في الملف MID، مثل صوت الكلام البشري أو الغناء.

الملفات الحركية AVI

تخزن هذه الملفات صوراً متتابعة، يتم عرضها بشكل متتالي لتشكل فيلماً متحركاً. في الحقيقة فإن هذه الملفات قادرة على تخزين الصوت أيضاً مع الصور المتحركة. وتستطيع تخيل هذه الملفات على أنها جهاز فيديو منزلي.

تأخذ هذه الملفات حجماً كبيراً جداً على وسائط التخزين، يعتمد الحجم على دقة الصور التي تخزنها، بالإضافة لعدد الألوان وجودة الصوت المخزن مع الصورة المتحركة.

هناك تنسيقات مختلفة جداً لهذا الملف منها:

تنسيق 15 صورة بالثانية الواحدة، أو 24 صورة بالثانية الواحدة، أو 30 صورة بالثانية الواحدة.

كلما زادت عدد الصور بالثانية الواحدة، زاد حجم الملف النهائي.

تعرفنا مما سبق على بعض أهم أنواع ملفات الوسائط المتعددة القياسية وكيفية تخزين المعلومات بداخلها. وأنشأت برنامجاً يستطيع عزف هذه الملفات للمستخدم.

الخلاصة

تعلمت في هذا الفصل عن تقنية OCX ActiveX، وكيف تتعامل مع كائنات ActiveX المختلفة. وتعلمت أيضاً قواعد استخدام هذه العناصر، وكيفية إضافة أحد هذه العناصر إلى مشروعك، لتتمكن من استخدامها.

تعلمت أيضاً كيفية بناء برنامج الوسائط المتعددة، الذي يتيح لك إمكانية فتح الملفات

القياسية للوسائط المتعددة، مثل ملفات الصوت Wave، والموسيقى MIDI، والحركة AVI، وأيضاً تشغيل القرص الليزري الموسيقى.
أخيراً تعلمت عن الملفات القياسية للوسائط المتعددة بشكل أعمق وكيفية تخزين المعلومات بداخلها وتنسيقاتها المختلفة وأحجامها التقريبية.

الفصل العشرون

استخدام توابع النظام Windows API

سوف تتعلم في هذا الفصل كيفية استدعاء توابع النظام API من داخل برامج فيجول بيسك، تتيح لك هذه التقنية استخدام قوة وسرعة وشمولية هذه التوابع، مباشرة من برنامجك في فيجول بيسك.

توابع فيجول بيسك القياسية

لقد استخدمت فعلياً العديد من توابع فيجول بيسك خلال قراءتك هذا الكتاب، مثال على ذلك التابع Str() الذي يحول المتحول الرقمي إلى متحول نصي، والتابع MsgBox() الذي يُظهر رسالة للمستخدم بأزرار معينة، ومن ثم يعود بقيمة تمثل رد المستخدم على هذه الرسالة.

ماذا لو كانت هناك بعض المهام التي تريد تأديتها، غير مدعومة أو موجودة في لغة فيجول بيسك؟ في هذه الحالة، قد تجد دعماً لهذه المهمة مع توابع النظام ويندوز API، المشروحة لاحقاً في هذا الفصل؟.

مكتبات الربط الديناميكية

يمكن أن يكون امتداد ملف مكتبة الربط الديناميكية، الامتداد DLL (مثل الملف MyFile.DLL)، أو أن يكون له الامتداد EXE (مثل الملف MyFile.EXE). والأشهر هو النوع الأول، لذلك أخذ الأحرف الأولى من اسمها Dynamic Link Libraries.

تحتوي ملفات مكتبات الربط الديناميكية DLL، على توابع داخلها، يمكن استدعاؤها من أي برنامج آخر، وكأنها جزء من هذا البرنامج.

دعنا نفترض وجود ملف DLL معين، يتضمن التابع المسمى MyFunction()، ولنفترض أيضاً أن هذا التابع يتطلب وسيطاً واحداً، ويعود بقيمة معينة. وهو مشابه للتابع Str() الموجود ضمناً في فيجول بيسك، ويتطلب وسيطاً واحداً هو قيمة رقمية ما، ويعود بقيمة نصية String تمثل هذا الرقم.

تستطيع الآن، كتابة برنامج فيجول بيسك يستخدم هذا التابع MyFunction()، الموجود في الملف MyFile.DLL، في هذه الحالة يستطيع برنامجك استخدام هذا التابع، بنفس الطريقة التي تستخدم فيها أي تابع داخلي في فيجول بيسك.

بمعنى آخر، تستطيع زيادة عدد التوابع التي يمكنك استخدامها في فيجول بيسك، بواسطة تقنية مكتبات الربط الديناميكية DLL.

تعتبر الميزة الرائعة لملفات الربط الديناميكية، هي في إمكانية استخدامها من قبل أكثر من برنامج وبنفس الوقت، مختصرة بذلك الكثير من التكرار الغير ضروري، والمستهلك لمساحة القرص الصلب.

يمكن كتابة برنامج فيجول بيسك، يستدعي تابعاً معيناً من مكتبة DLL ما، يقوم هذا البرنامج خلال عملية تنفيذه، بتحميل الجزء الحاوي على شفرة التابع المستدعى، والموجودة في ملف مكتبة DLL، يقوم بتحميله في الذاكرة ومن ثم ينفذه.

يمكنك الآن فهم سبب تسمية هذه الملفات بمكتبات الربط الديناميكية.

هذه الملفات عبارة عن مكتبات من التوابع، تحمّل هذه المكتبات (ربط هذه المكتبات) مع برنامجك حسب الحاجة فقط (ديناميكياً)، وهي ليست جزءاً من ملف برنامجك التنفيذي وإنما تتواجد في ملف منفصل عنه.

علاوةً على ذلك، يمكنك كتابة برنامج فيجول بيسك آخر، يستخدم نفس التابع الموجود في مكتبة DLL (أو تابع آخر موجود في نفس المكتبة). عند تنفيذ هذا البرنامج (والبرنامج الأول مازال منفذاً)، يصبح لديك برنامجان يقومان باستدعاء نفس التابع الموجود في الملف DLL نفسه. وفي الواقع، يمكن وجود أكثر من برنامج يقوم بالاستفادة من نفس مكتبة DLL.

بكلام ملخص جداً، تعتبر مكتبات DLL، ملفات تحوي توابع بداخلها، وهي متاحة للاستخدام من أي برنامج وفي نفس الوقت.

توابع النظام ويندوز API

قد تلاحظ (كمشغل لويندوز)، وجود العديد من المزايا المتوفرة في النظام ويندوز، مثل تحريك الفأرة، نقر الفأرة، اختيار البنود من القوائم، الخ.

بالطبع، يستطيع ويندوز تنفيذ العديد من المهام الأخرى، مثل حفظ الملفات، إظهار الصور، إدارة أجهزة الكومبيوتر المختلفة، وأداء الآلاف من العمليات الهامة الأخرى.

يعتبر ويندوز بحد ذاته برنامجاً، مثل بقية البرامج الأخرى، ولكن بالنسبة للمستخدمين، فإنهم لا يميلون لاعتباره برنامجاً، بل يعتبرونه آليةً تمكنهم من تنفيذ البرامج الأخرى (وآليةً لتطوير برامج، بواسطة لغات البرمجة مثل فيجول بيسك).

بكلمة أصح، معظم الأعمال التي ينفذها ويندوز، هي في الحقيقة استدعاءات للتوابع الموجودة في ملفات مكتبات DLL.

وعندما يريد ويندوز أداء مهمة معينة، فإنه فعلياً يستدعي التابع الخاص بهذه المهمة من خلال ملف مكتبة DLL المحتوية على هذا التابع.

لقد رأيت مسبقاً كيف أن العديد من البرامج يمكنها استخدام نفس التابع من الملف DLL نفسه وبشكل آني وهذا يعني أن برنامجك المطور في لغة فيجول بيسك يمكنه استخدام التوابع الموجودة في نفس ملف DLL والذي يستخدمه ويندوز نفسه.

والآن دعنا نرى الفوائد من استخدام التوابع التي يستخدمها ويندوز نفسه:

- تعتبر مكتبات DLL التي يستخدمها ويندوز موجودة في جهازك الشخصي فعلياً، ويفترض أن مستخدمي برنامجك لديهم ويندوز أيضاً وهذا يعني أنه لا حاجة لتوزيع ملفات DLL الخاصة بويندوز (وهي كبيرة جداً) مع ملفات برنامجك، لأن ملفات الويندوز موجودة مسبقاً عل أجهزة باقي المستخدمين.
- تعمل توابع مكتبات DLL بشكل جيد وخالي من الأخطاء وبذلك تضمن الثقه فى برامجك وتوافقيتها على جميع أجهزة الكمبيوتر الشخصية التي تستخدم النظام ويندوز.
- لا توجد لغة برمجة يمكنها تنفيذ كل المهام التي يمكن للنظام ويندوز أن يؤديها، حتى أعقد اللغات مثل فيجول سي++، لذلك لابد لك من استخدام توابع النظام ويندوز API بشكل مباشر.
- يوجد العديد من التوابع الخاصة بالنظام ككل، وعند استدعاء أحد هذه التوابع من مكتبات ويندوز نفسه، تضمن وثوقية عمل النظام بشكل كامل.

لنفرض مثلاً احتياج برنامجك في مرحلة ما، لإعادة إقلاع الجهاز Reset. في حال اعتمدت على تابع إعادة الإقلاع الخاص بك، توقع أن يسبب تابعك بعض

المشاكل.

مثلاً، هناك برنامج منفذ حالياً، ويحتاج لحفظ آخر التعديلات التي جرت على ملف ما، أو هناك مستخدمين آخرين متصلين مع جهازك الشخصي، وإعادة إقلاع الجهاز بدون تنبيه وتنفيذ بعض المهام قبل عملية إعادة الإقلاع، يتسبب في الكثير من الضرر.

أما في حال استخدمت ما إذا تابع إعادة الإقلاع الخاص بالنظام ويندوز، فتأكد بأنه سيقوم بالعمل بدون أي أضرار، وسيقوم بجميع الإجراءات الضرورية، مثل تنبيه باقي البرامج أو المستخدمين على ضرورة الخروج حالياً، ويمكن في بعض الأحيان إلغاء أمر إعادة الإقلاع لأسباب خاصة.

لقد رأيت الآن، وجود أسباب كثيرة ومنطقية، لاستخدام توابع الويندوز API، لنكتب الآن برنامجاً يستخدم توابع النظام ويندوز، ونرى كيفية عمله.

ملاحظة

صممت مايكروسوفت النظام ويندوز، بطريقة تسمح له باستخدام التوابع من ملفات DLL. بكلمة أصح، صممت مايكروسوفت هذه المكتبات بطريقة تسمح للبرامج الأخرى (مثل البرامج المطورة في فيجول بيسك)، بقابليتها لاستخدام هذه التوابع، من ملفات DLL الخاصة بويندوز. تسمى توابع النظام ويندوز (واجهة برمجة التطبيقات) ويطلق عليها اختصاراً API (Application Programming Interface).

برنامج API

سننشئ الآن، البرنامج API، يوضح هذا البرنامج، كيفية استخدام توابع API من خلال برامجك المطورة في فيجول بيسك.

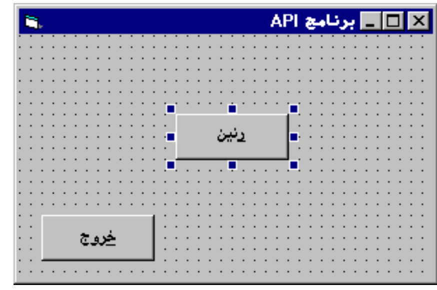
- أنشئ الدليل C:\VB5Prg\Ch20، لكي تحفظ المشروع فيه.
- أنشئ مشروعاً جديداً بنوع Standard EXE، من القائمة **File**.
- اختر البند **Save Form1 As** من القائمة **File**، ثم احفظه باسم MyApi.Frm، في نفس الدليل.

- اختر البند **Save Project As** من القائمة **File**، ثم احفظه باسم MyApi.Vbp في نفس الدليل السابق أيضاً.
- صمم النموذج frmMyApi وفق الجدول 20-1.

بعد انتهائك من بناء النموذج frmMyApi، ينبغي ظهوره كالشكل 20-1.

الجدول 20-1. جدول خصائص النموذج frmMyApi.

الكائن	الخاصية	القيمة
Form	**Name**	**frmMyApi**
	Caption	برنامج API
	RightToLeft	True
CommandButton	**Name**	**cmdBeep**
	Caption	&رنين
CommandButton	**Name**	**cmdExit**
	Caption	&خروج



الشكل 20-1
النموذج frmMyApi
بعد انتهاء تصميمه.

- أضف النص التالي لقسم التصريحات العامة للنموذج frmMyApi:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

- أضف النص التالي لحادثة Click الخاصة بالزر cmdExit:

```
Private Sub cmdExit_Click()
    End
```

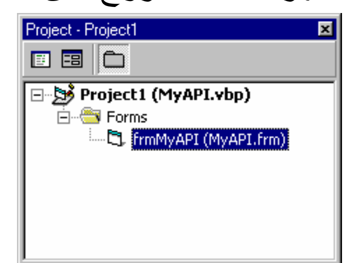
```
End Sub
```

يتسبب نقر الزر **خروج**، بإنهاء البرنامج والعودة إلى فيجول بيسك.

إضافة وحدة نمطية BAS جديدة للمشروع

تحتوي نافذة المشروع على نموذج واحد فقط هو frmMyApi، (انظر الشكل 20-2 الممثل لنافذة المشروع).

أظهر نافذة المشروع، من القائمة **View**، البند **Project Explorer**.



الشكل 20-2

يحتوي المشروع MyApi على نموذج وحيد.

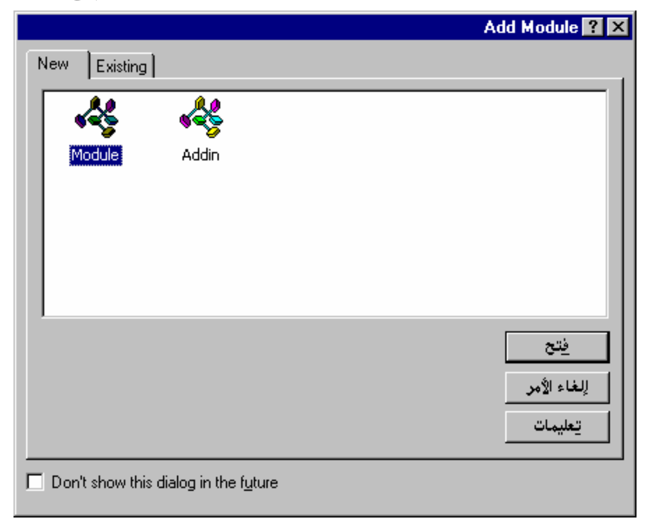
سنضيف وحدة نمطية جديدة BAS إلى المشروع MyApi:

- اختر البند **Add Module**، من القائمة **Project**.

يستجيب فيجول بيسك بإظهار نافذة Add Module (انظر الشكل 20-3).

- تأكد أن صفحة New هي الظاهرة في نافذة Add Module.
- اختر الرمز Module، ثم انقر الزر **فتح**.

نتيجة ذلك، تضاف وحدة نمطية جديدة BAS إلى المشروع.



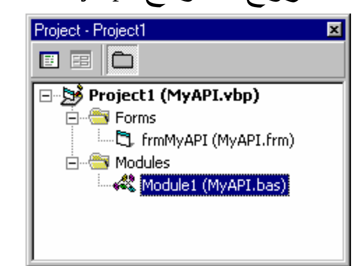
الشكل 20-3

إضافة وحدة نمطية جديدة للمشروع MyApi.Vbp.

للوحدة النمطية المضافة اسم افتراضي هو Module1، لذلك يجب تغيير الاسم إلى MyApi.BAS كالتالي:

- اختر البند **Save Module1 As** من القائمة **File**، واحفظه باسم MyApi.BAS في الدليل C:\VB5Prg\Ch20.

انظر لنافذة المشروع مرة أخرى، وكما ترى (انظر الشكل 20-4)، تتضمن نافذة المشروع النموذج frmMyApi، والوحدة النمطية MyApi.BAS.



الشكل 20-4

تتضمن نافذة المشروع النموذج frmMyApi، والوحدة النمطية MyApi.BAS.

التصريح عن توابع API

هدفنا من هذا البرنامج، تنفيذ تابع API من خلال البرنامج API، لذلك يجب إخبار فيجول بيسك عن اسم التابع التي تود استخدامه، وأين يوجد (اسم الملف DLL الذي يحويه)، وكيفية عمل هذا التابع (الوسائط المطلوبة لهذا التابع والقيمة العائدة منه).

تدعى عملية إخبار فيجول بيسك عن تفاصيل التابع بالتصريح Declare عن هذا التابع.

أضف الأسطر التالية إلى قسم التصريحات العامة للوحدة النمطية MyApi.BAS:

```
'يجب التصريح عن كل المتحولات
Option Explicit
Declare Function MessageBeep Lib "User32" _
                          (ByVal wType As Long) As Long
```

- اختر البند **Save Project** من القائمة **File**، لحفظ المشروع كاملاً.

يعتبر السطر التالي في قسم التصريحات العامة للوحدة النمطية MyApi.BAS، سطر التصريح عن تابع API:

```
Declare Function MessageBeep Lib "User32" _
                          (ByVal wType As Long) As Long
```

يبدأ سطر التصريح عن تابع API بالعبارة Declare، بعد Declare مباشرة، يأتي دور تحديد نوع التابع، هل يعود التابع بقيمة (Function)، أم لا يعود بقيمة (Sub).

بعد تحديد نوع التابع، ينبغي تحديد اسم التابع المراد استدعاؤه. في مثالنا الحالي، يسمى التابع MessageBeep.

بعد تحديد اسم التابع، ينبغي تحديد موقعه. في مثالنا الحالي، التابع موجود في المكتبة Lib المسماة User32.

يتواجد الملف User32.DLL، في الدليل الفرعي System، الموجود في الدليل الرئيسي C:\Windows، (بالنسبة للنظام Windows95).

ويتواجد في الدليل C:\WinNT\System32، (بالنسبة للنظام WinNT).

علِم فيجول بيسك نتيجة سطر التصريح السابق، موقع التابع MessageBeep()، (المكتبة User32.DLL).

بعد تحديد موقع التابع، ينبغي تحديد الوسائط المطلوبة لعمله، ونوع البيانات الخاصة بكل وسيط. يتطلب التابع MessageBeep() وسيطاً واحداً فقط، هو wType من النوع Long.

أخيراً، بعد تحديد كل الوسائط المطلوبة (بشكل عام)، يأتي دور تحديد نوع القيمة التي يعود بها التابع بعد انتهاء مهمته. يعود التابع MessageBeep() بقيمة من النوع Long، لذلك كتبت العبارة As Long آخر سطر التصريح.

ستتعلم في آخر هذا الفصل، كيفية إيجاد أسطر التصريحات الخاصة بتوابع API الأخرى.

تنفيذ التابع MessageBeep()

لنكتب الآن العبارات الخاصة بتنفيذ التابع MessageBeep:

يجب ملاحظة أمر هام جداً، هو كون عبارة تنفيذ التابع، مطابقة تماماً لما جاء في سطر التصريح عن هذا التابع، من حيث الوسائط المطلوبة ونوع كل وسيط، ونوع القيمة العائدة من التابع.

□اكتب الأسطر التالية في الإجراء ()cmdBeep_Click:

```
Private Sub cmdBeep_Click()
    Dim Dummy
    Dummy = MessageBeep(1)
End Sub
```

□اختر البند **Save Project** من القائمة **File**، لحفظ المشروع كاملاً.

صرحت في الإجراء السابق عن متحول محلي باسم Dummy كالتالي:

```
    Dim Dummy
```

ثم استدعيت التابع ()MessageBeep:

```
    Dummy = MessageBeep(1)
```

وأسندت القيمة العائدة من التابع ()MessageBeep إلى المتحول Dummy.

فعلياً، وفي هذا المثال بالذات، لا فائدة أبداً من القيمة العائدة من هذا التابع، ولكن فقط لتوضيح أن تابع API يعود بقيمة، يجب إسنادها إلى متحول ما.

يحتاج التابع ()MessageBeep إلى وسيط واحد فقط، ويعرِّف هذا الوسيط كيفية إصدار الصوت بالضبط، كما سيتم شرحه لاحقاً.

□نفّذ برنامج API.

□انقر الزر **رنين**، وتأكد من سماعك صوتاً.

□تمرن على البرنامج، ثم انقر الزر **خروج** لإنهاء البرنامج.

بالتأكيد، يمكن استخدام العبارة Beep الجاهزة في فيجول بيسك، بدلاً من هذه الطريقة الطويلة لعمل نفس الشيء. لكن الهدف من هذا التمرين، هو معرفة كيفية استدعاء تابع API ما.

تعتمد طريقة إصدار الصوت، على كيفية تعريف بطاقة الصوت لديك. قد يصدر الصوت من خلال بطاقة الصوت، وليس من خلال سماعة الجهاز الداخلية.

غيِّر قيمة الوسيط المطلوب للتابع ()MessageBeep من 1 إلى −1 وذلك كما يلي:

```
    Dummy = MessageBeep(-1)
```

□نفّذ البرنامج مرة أخرى، وتأكد من إصدار الصوت عبر سماعة الجهاز هذه المرة. تُجبر القيمة −1، التابع ()MessageBeep على إصدار الصوت عبر سماعة

الجهاز الداخلية، حتى لو كان لديك بطاقة صوت معرفة بشكل صحيح.

عند إصدار الصوت عبر سماعة الجهاز الداخلية، تكون فترة إصدار الصوت صغيرة جداً (بالكاد تسمعها). لكي تطيل فترة إصدار الصوت، غيِّر الإجراء cmdBeep_Click() إلى:

```
Private Sub cmdBeep_Click()
    Dim Dummy
    Dim I
    For I=0 To 100
      Dummy = MessageBeep(1)
    Next
End Sub
```

تتسبب إضافة الحلقة For-Next في تنفيذ التابع مائة مرة متتالية.

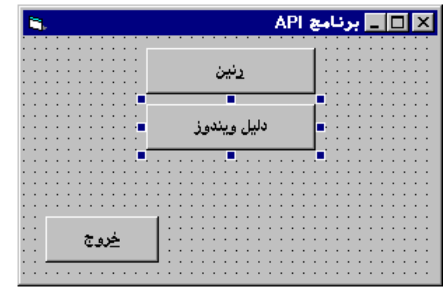
معرفة اسم دليل Windows

كمثال على استدعاء تابع API آخر، دعنا نستخدم تابع API، الذي يخبرنا عن اسم الدليل الذي جُهِّز فيه النظام Windows:

- ضع زراً جديداً على النموذج frmMyApi.
- أسند القيم التالية لخصائصه:

```
Name:    cmdWhereWindows
Caption: دليل ويندوز
```

ينبغي ظهور النموذج بعد الانتهاء من تصميمه، كما في الشكل 20-5.



الشكل 20-5

النموذج frmMyApi

بعد إضافة الزر **دليل ويندوز**.

- أضف الأسطر التالية إلى قسم التصريحات العامة للوحدة النمطية MyApi.Bas.

بعد الانتهاء من إضافة الأسطر الجديدة، يصبح كالتالي:

```
'يجب التصريح عن كل المتحولات
Option Explicit
```

```
Declare Function MessageBeep  Lib "User32" _
                      (ByVal wType As Long) As Long
Declare Function GetWindowsDirectory Lib "Kernel32" _
    (ByVal lpBuffer As String,ByVal nSize As Long) As Long
```

لاحظ، أن التصريح الجديد عن التابع GetWindowsDirectory()، أصعب وأعقد قليلاً من التصريح عن التابع السابق MessageBeep().

اسم التابع الثاني GetWindowsDirectory()، وهو موجود في ملف المكتبة Kernel32.Dll.

ظهر قسم جديد في سطر التصريح الثاني، هو العبارة Alias. حيث يمكننا تغيير اسم التابع الأصلي إلى اسم مستعار آخر، وذلك في حال حدوث تعارض بين أسماء التوابع مع بعضها البعض، أو مع عبارات فيجول بيسك المحجوزة الأخرى. يكفينا ما تعلمناه عن العبارة Alias، ولن نخوض في تفاصيلها أكثر من ذلك.

يتطلب التابع GetWindowsDirectory() وسيطين هما: الوسيط lpBuffer من النوع String، والوسيط nSize من النوع Long:

```
(ByVal lpBuffer As String,ByVal nSize As Long)
```

يعود التابع بقيمة بعد تنفيذه، من النوع Long.

- اختر البند **Save Project** من القائمة **File**، لحفظ المشروع كاملاً.

ربط نص البرنامج الخاص بحادثة Click للزر دليل ويندوز

اتبع الخطوات التالية:

- اكتب الأسطر التالية في الإجراء cmdWhereWindows_Click():

```
Private Sub cmdWhereWindows_Click ()
    Dim Result
    Dim WindowsDirectory A String
    WindowsDirectory = Space(144)
    Result = GetWindowsDirectory(WindowsDirectory,144)
    If Result = 0  Then
      MsgBox "لم أستطع الحصول على اسم مجلد ويندوز"
    Else
      WindowsDirectory = Trim(WindowsDirectory)
      MsgBox "مجلد ويندوز هو:" & WindowsDirectory
    End If
```

```
End If
```

□اختر البند **Save Project** من القائمة **File**، لحفظ المشروع كاملاً.

صرحت في الأسطر السابقة عن متحولين هما:

```
Dim Result
Dim WindowsDirectory A String
```

ثم ملأت المتحول WindowsDirectory، بأحرف مسافات (144 حرف مسافة):

```
WindowsDirectory = Space(144)
```

يغنينا التابع ()Space، عن كتابة أحرف المسافات فعلياً، لإسنادها للمتحول. ولولا هذا التابع، لاضطررنا لكتابة السطر التالي:

```
WindowsDirectory = " (اضغط مفتاح المسافة 144 مرة هنا) "
```

بعد ذلك، نفّذ التابع ()GetWindowsDirectory كالتالي:

```
Result = GetWindowsDirectory(WindowsDirectory,144)
```

تُسند النتيجة (القيمة العائدة من التابع) للمتحول Result. لا يحتوي المتحول Result على اسم مجلد النظام Windows، بل يحتوي على رقم، يمثل نجاح التابع في أداء عمله أو فشله.

إذا كانت قيمة المتحول Result مساوية للصفر، يكون التابع قد فشل في أداء مهمته ولسبب من الأسباب، أما إذا كانت قيمة المتحول Result لا تساوي الصفر، يكون التابع قد نجح في أداء مهمته.

تساوي قيمة الوسيط الثاني 144، وهي تمثل طول سلسلة الأحرف التي ينبغي وضعها في المتحول WindowsDirectory، يُستخدم المتحول WindowsDirectory كخرج Output لمعلومات التابع ()GetWindowsDirectory.

بكلام آخر، يضع التابع ()GetWindowsDirectory اسم مجلد الويندوز في المتحول WindowsDirectory. من الضروري جداً ملء المتحول WindowsDirectory بأحرف مسافات وبطول 144 حرف، قبل استدعاء التابع ()GetWindowsDirectory، لأن التابع يُحدِّث سلسلة الأحرف الموجودة في المتحول WindowsDirectory، ويفترض هذا التابع وجود منطقة من الذاكرة، لوضع سلسلة الأحرف الجديدة (اسم مجلد ويندوز)، قبل عملية تنفيذه.

بعد ذلك، نفذت العبارة If-Else-End If التالية:

```
If Result = 0  Then
  MsgBox "لم أستطع الحصول على اسم مجلد ويندوز"
Else
  WindowsDirectory = Trim(WindowsDirectory)
  MsgBox "مجلد ويندوز هو:" & WindowsDirectory
End If
```

إذا كانت قيمة Result تساوي الصفر (القيمة العائدة من التابع)، هذا يعني فشل التابع GetWindowsDirectory() في الحصول على اسم الدليل الذي جُهِّز فيه النظام Windows لسبب من الأسباب. ينبغي على المبرمج، توضيح هذا الأمر للمستخدم، وإظهار رسالة له، تخبره عن عدم قدرته في الحصول على اسم مجلد الويندوز.

أما إذا كانت قيمة المتحول Result لا تساوي الصفر، هذا يعني نجاح التابع في الحصول على اسم الدليل، وتنفذ الأسطر الواقعة بعد العبارة Else. وهي:

```
WindowsDirectory = Trim(WindowsDirectory)
MsgBox "مجلد ويندوز هو:" & WindowsDirectory
```

تكون قيمة المتحول WindowsDirectory مبدئياً، سلسلة من المسافات بطول 144 حرف مسافة. لذلك اضطررنا لاستخدام التابع Trim() لجعل المتحول WindowsDirectory خالياً من المسافات الزائدة.

ملاحظة

يزيل التابع Trim() أحرف المسافات الزائدة من يمين المتحول ويساره، لكنه لا يزيل أحرف المسافات الفاصلة بين الكلمات الموجودة في المتحول.

مثلاً، لو كان لدينا العبارات التالية:

```
myName = "     Ahmad  Waddah      "
myName = Trim(myName)
Print myName
```

بعد تنفيذها، تكون النتيجة:

```
Ahmad  Waddah
```

نلخص الكلام السابق فنقول:

▪هيأت المتحول WindowsDirectory لاستقبال المعلومات من التابع

.GetWindowsDirectory()

- أرسلت هذا المتحول للتابع عن طريق استدعاء التابع فعلياً.
- وَضع التابع اسم مجلد Windows في المتحول WindowsDirectory.
- غير التابع GetWindowsDirectory() قيمة المتحول WindowsDirectory من سلسلة أحرف مسافات، إلى سلسلة أحرف تمثل اسم الدليل، مع بقاء المسافات الزائدة في آخر المتحول.
- حَذَف التابع Trim() المسافات الزائدة من المتحول WindowsDirectory.
- أظهرت العبارة MsgBox اسم الدليل، للمستخدم.

- احفظ النموذج بضغط مفتاحي Ctrl + S.
- نفذ البرنامج وتأكد من ظهور رسالة تخبرك عن اسم مجلد Windows، عند نقر الزر **دليل ويندوز**.

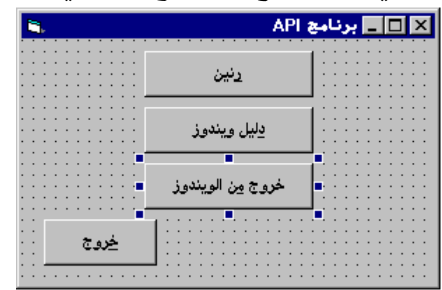
إضافة زر الخروج من الويندوز

اتبع ما يلي:

- ضع زراً جديداً على النموذج frmMyApi.
- أسند القيم التالية لخصائصه:

```
Name:    cmdExitWindows
Caption: خروج &من الويندوز
```

ينبغي أن يصبح النموذج كما في الشكل 20-6.



الشكل 20-6

النموذج frmMyApi بعد

إضافة زر الخروج من الويندوز.

أين توجد أسطر التصريح عن توابع API

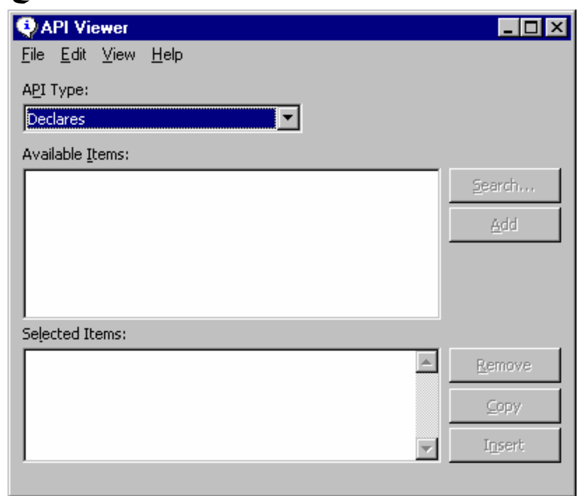
رأيت سابقاً، كلاً من التابع MessageBeep()، والتابع GetWindowsDirectory()، وعرفت كيفية استخدامهما عن طريق هذا الكتاب. تصور أنك تريد الآن، استخدام تابع يتسبب في إعادة إقلاع الجهاز Reboot.

كيف تعرف اسم التابع API الصحيح؟ وما هو السطر الذي يصرح عنه بشكل صحيح؟ أين أجد هذه المعلومات؟.

انظر إلى دليل فيجول بيسك، ستجد مجلداً فرعياً يسمى Winapi\، والملف التنفيذي Apiload.exe.

- نفّذ البرنامج Apiload.exe عن طريق مستكشف Windows، أو عن طريق قائمة زر ابدأ، ثم بند البرامج، ثم بند **Microsoft Visual Basic**، ثم بند **API Text Viewer**، وهو أحد بنود المجموعة Microsoft Visual Basic.

تظهر نافذة API Viewer نتيجة تشغيل البرنامج، (الشكل 20-7).



الشكل 20-7

نافذة API Viewer.

استخدم النافذة السابقة، لمعرفة كيفية التصريح عن تابع معين. كالتالي:

- اختر البند **Load Text File** من قائمة **File**، في البرنامج API Viewer.

يظهر مربع الحوار Select a Text API File.

- اختر الملف Win32api.txt من الدليل الفرعي Winapi، ثم انقر **فتح**.

تُحمَّل جميع البنود الموجودة في الملف Win32api.txt، إلى مربع السرد Available

Items.

□تأكد من اختيار البند Declare في الحقلAPI Type.

□يتضمن مربع السرد Available Items الآن، جميع تصريحات توابع النظامAPI.

□ابحث عن البند ExitWindowsEx، وانقره مرة واحدة لاختياره.

□انقر الزر **Add** الموجود في نافذة البرنامج API Viewer.

يظهر البند المضاف في الحقل Selected Items (أسفل نافذة البرنامج).

□استخدم الفأرة لاختيار محتويات الحقل Selected Items، ثم انقر الزر Copy.

يتم نسخ محتويات الحقل Selected Items المختارة، إلى حافظة النظام Windows.

□ضع مؤشر الفأرة على قسم التصريحات العامة للوحدة النمطية MyApi.Bas، وذلك بنقر موقع الكتابة (لتنشيط نافذة نص البرنامج)، ثم انتقل إلى آخر سطر.

□الصق النص الموجود في حافظة Windows، (وذلك بضغط المفتاحين Ctrl+V).

يصبح النص الموجود في قسم التصريحات العامة للوحدة النمطية MyApi.Bas كما يلي:

```
'يجب التصريح عن كل المتحولات
Option Explicit
Declare Function MessageBeep  Lib "User32" _
                      (ByVal wType As Long) As Long
Declare Function GetWindowsDirectory Lib "Kernel32" _
    (ByVal lpBuffer As String,ByVal nSize As Long) As Long
Declare Function ExitWindowsEx Lib "User32" _
   (ByVal uFlags As Long,ByVal dwReserved As Long) As Long
```

تحتاج من برنامج API Viewer أيضاً، الثوابت المستخدمة مع توابع API.

□عد ثانية إلى البرنامج API Viewer، واختر البند Constant من الحقل API Type (حتى تظهر لائحة بثوابت توابع API).

قد يظهر لك مربع حوار، يسألك إذا كنت تريد تحويل الملف المحمَّل الحالي، إلى ملف Database، جاوب بنعم، فيظهر مربع الحوار Select a Name for New Database.

□احفظ ملف قاعدة البيانات باسم Win32api.Mdb في الدليل Winapi\.

تأخذ عملية التحويل بعض الوقت، ولكن عند تنفيذ البرنامج API Viewer في المرة المقبلة، اختر البند **Load Database File**، واختر الملف Win32api.Mdb.

يتعامل البرنامج في هذه الحالة، مع ملف قاعدة بيانات، وليس مع ملف نصي، وستلاحظ الفرق الكبير في السرعة التي يتعامل فيها مع البنود من حيث العرض أو البحث الخ.

في جميع الحالات، تظهر الآن جميع ثوابت التوابع API.

- أزح البنود عن طريق الأسهم أو شريط التمرير الأفقي، حتى ترى البند EWX_SHUTDOWN، انقر عليه مرة واحدة فقط، ثم انقر الزر **Add**.

يظهر البند EWX_SHUTDOWN في المربع السفلي من البرنامج API Viewer.

- اختر البند نفسه من المربع السفلي، ثم انقر الزر **Copy**.

يتم نسخ هذا البند إلى الحافظة Clipboard.

- انقل مؤشر الفأرة إلى آخر سطر في الوحدة النمطية MyApi.Bas، ثم الصق النص الموجود في الحافظة.

يصبح الآن النص الموجود في قسم التصريحات العامة للوحدة النمطية MyApi.Bas كالتالي:

```
'يجب التصريح عن كل المتحولات
Option Explicit
Declare Function MessageBeep  Lib "User32" _
                      (ByVal wType As Long) As Long
Declare Function GetWindowsDirectory Lib "Kernel32" _
    (ByVal lpBuffer As String,ByVal nSize As Long) As Long
Declare Function ExitWindowsEx Lib "User32" _
   (ByVal uFlags As Long,ByVal dwReserved As Long) As Long
Public Const EWX_SHUTDOWN = 1
```

- اختر البند **Save Project** من القائمة **File**، لحفظ المشروع كاملاً.

إسناد نص برنامج حادثة Click للزر cmdExitWindows

اتبع ما يلي:

اكتب ما يلي في الإجراء ()cmdExitWindows_Click:

```
Private Sub cmdExitWindows_Click()
    Dim Dummy
    Dim Answer
    Answer = MsgBox("هل تريد الخروج من الويندوز بالتأكيد؟", _
                    vbYesNo)
    If Answer = vbYes Then
       Dummy = ExitWindowsEx(EXW_SHUTDOWN,0)
    End If
End Sub
```

يصرح نص البرنامج الذي كتبته سابقاً، عن متحولين محليين هما: Dummy و Answer. لكتابة برنامج احترافي ووثوقي، يجب التأكد أن المستخدم يريد وبشكل مؤكد، تنفيذ العمل الذي طلبه من البرنامج.

في هذا المثال، تم التأكد من نية المستخدم على الخروج، قبل تنفيذ تابع الخروج من الويندوز، بإظهار رسالة واضحة قابلة للتراجع (نقر الزر **لا**).

إذا نقر المستخدم زر **نعم**، ينفذ السطر الذي يلي تعليمة If:

```
Dummy = ExitWindowsEx(EXW_SHUTDOWN,0)
```

لاحظ عدم استخدام القيمة العائدة من التابع (لا تهمك حالياً)، لذلك تم وضعها في متحول باسم Dummy (زائف).

يغلق التابع النظام Windows بكامله، بسبب وضع قيمة الوسيط الأول مساوية لقيمة الثابت EXW_SHUTDOWN.

بالعودة إلى قسم التصريحات العامة للوحدة النمطية MyApi.Bas، نجد السطر التالي:

```
Public Const EXW_SHUTDOWN = 1
```

يدل هذا السطر على أن قيمة الثابت EXW_SHUTDOWN تساوي الواحد، وهي قيمة ثابتة لا يجوز تغيرها ضمن البرنامج.

إذاً، السطرين التاليان متكافئان من حيث النتيجة:

```
Dummy = ExitWindowsEx(EXW_SHUTDOWN,0)
Dummy = ExitWindowsEx(1,0)
```

تستخدم تقنية تعريف الثوابت، لجعل البرنامج أوضح في الفهم، ومن البديهي أن السطر الأول أوضح من السطر الثاني.

قبل تجريب البرنامج API، تأكد من حفظ المشروع، وإغلاق كافة التطبيقات الأخرى.

- نفّذ البرنامج API.
- انقر زر **خروج من الويندوز**.

تظهر رسالة تأكيد، تطلب منك إجابة صريحة بنعم أو لا.

- انقر نعم.
- تأكد من إغلاق النظام Windows.

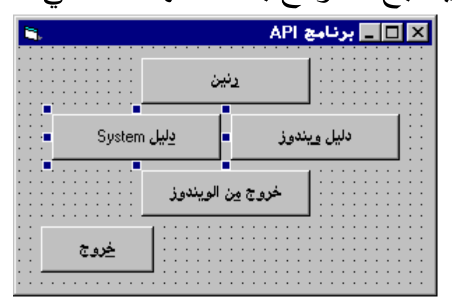
معرفة اسم مجلد Windows\System

بطريقة مشابهة جداً لمعرفة اسم مجلد Windows، يمكننا معرفة اسم مجلد Windows\System، اتبع ما يلي:

- أضف زراً جديداً للنموذج frmMyApi.
- أسند القيم التالية لخصائصه:

```
Name:    cmdWhereSystem
Caption: System &دليل
```

يصبح النموذج بعد الانتهاء كما في الشكل 20-8.



الشكل 20-8

النموذج بعد إضافة زر دليل System.

- أضف سطر التصريح التالي لقسم التصريحات العامة للوحدة النمطية MyApi.Bas:

```
Declare Function GetSystemDirectory Lib "kernel32" Alias _
                "GetSystemDirectoryA" _
                (ByVal lpBuffer As String, _
                 ByVal nSize As Long) As Long
```

◻أضف الأسطر التالية للإجراء cmdWhereSystem_Click():

```
Private Sub cmdWhereSystem_Click()
    Dim Result
    Dim SystemDirectory A String

    SystemDirectory = Space(144)

    Result = GetSystemDirectory(SystemDirectory,144)

    If Result = 0  Then
      MsgBox "لا يمكن الحصول على اسم مجلد النظام"
    Else
      MsgBox  "مجلد النظام هو:" & SystemDirectory
    End If
End Sub
```

من المؤكد أنك وجدت الأسطر السابقة، مشابهة جداً للأسطر التي كتبتها في إجراء الحصول على اسم دليل Windows، مع اختلاف بسيط جداً هو في اسم التابع API.

الخلاصة

تعلمت في هذا الفصل، كيفية استخدام توابع النظام API Windows من برامج فيجول بيسك. ورأيت أيضاً وجوب التصريح عن التابع قبل التمكن من استخدامه، وبمجرد التصريح عن التابع، يمكنك استخدامه وكأنه موجود فعلاً في فيجول بيسك. تزيد بهذه الطريقة عدد التوابع الممكن استخدامها.

شاهدت أيضاً برنامج API Viewer، الذي يساعدك على نسخ الأسطر المصرحة عن التوابع، ولصقها في برنامجك.

صحيح أن هذا البرنامج لا يعلمك طريقة استخدام التوابع، لكن يمكن في بعض الأحيان اكتشاف ما يفعله التابع من اسمه المجرد.

تتواجد المعلومات الكاملة عن توابع API، في ملفات التعليمات Help Files، والوثائق التي تأتي مع بعض اللغات الأخرى مثل لغة Visual C++.